Rekall and GRR

Searching for evil, together!

Michael Cohen
Google Inc

Rekall - What is it?

- A suite of Digital Forensic/Incident Response tools:
  - Pmem suite of memory acquisition tools.
    - MacPmem - only driver (commercial or opensource) that works with latest OSX.
    - WinPmem - the most advanced and reliable windows memory acquisition tool.
    - LinPmem - Linux acquisition driver (We usually use /proc/kcore though).
  - Live analysis capability:
    - Automatically load driver and access raw memory without taking an image first.
    - Used for triaging and collecting a more complete image.
    - Focus on automated acquisition with live analysis.
  - The largest repository for memory analysis profiles:
    - Rekall usually autodetects the right profile to use automatically.
  - Focus on user experience.
    - Automate as much as possible
    - Provide a complete working environment with a user friendly interface.

Installing Rekall

```
$ virtualenv  /path/to/MyEnv
New python executable in
/path/to/MyEnv/bin/python
Installing setuptools, pip...done.
$ source /path/to/MyEnv/bin/activate
$ pip install --upgrade setuptools pip wheel
$ pip install rekall
```

On Windows:

- Install Python MSI from python.org
- Install Microsoft compiler from http://aka.ms/vcpython27
- There is also a pyinstaller built single binary.

Workshop Notes

In this workshop we work in "Stand alone" mode.

- Because we don't want to rely on wireless.
- All dependencies were previously downloaded into a directory on the thumbdrive (wheelhouse):
  - `pip download rekall`
- We can install from this directory instead of the pypi index:
  - `set PIP_FIND_LINKS=XXX`
  - `set PIP_NO_INDEX=y`
  - `pip install rekall`

```
E:\>c:\Python27\Scripts\virtualenv.exe e:\dev
New python executable in e:\dev\Scripts\python.exe
Installing setuptools, pip, wheel...done.

E:\>e:\dev\Scripts\activate

(dev) E:\>set PIP_FIND_LINKS=e:\wheelhouse

(dev) E:\>set PIP_NO_INDEX=y

(dev) E:\>pip install rekall
Ignoring indexes: https://pypi.python.org/simple
```

PIP Crash course

- PIP is the python packaging tool.
  - `pip install rekall` - fetches Rekall and all its dependencies from the public index.
- Installing offline:
  - `pip download rekall` - downloads Rekall and all dependencies to the current directory then,
  - `pip install -f wheelhouse rekall` - fetches packages from local directory before the index.
  - `set PIP_FIND_LINKS=e:\wheelhouse` - Implicitly adds the -f flag to all pip commands.

Running Rekall for the first time.

- Rekall offers command line help
  - rekall --help
- For this exercise we run Rekall in live mode.
  - For this workshop we recommend using the **-v** flag to see what Rekall is doing under the covers.
  - There are two modes currently:
    - rekall -v --live API
    - rekall -v --live Memory

Working Offline

Rekall normally fetches the needed profile from the repository over the network.

What if I want to work offline or my internet connection sucks?

```
$ git clone
https://github.com/google/rekall-profiles.git
```

Or download the zip file from the github page:

```
https://github.com/google/rekall-profiles/archive/master
.zip
```

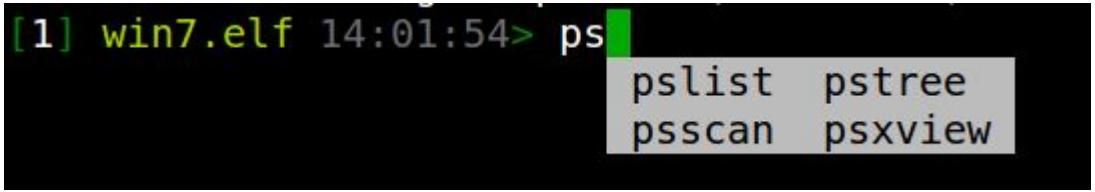
Edit ~/.rekallrc:

```
profile_path:
  - /path/to/my/rekall-profiles/
```

Note: For this workshop I have provided a copy of the main branch on the thumbdrive. Unzip it somewhere and point Rekall at it if the internet does not work for you. (`c:\python27\python -m zipfile -e foo.zip path`)

Rekall tips

Use command completion by typing tab twice



Follow the name of a command by ? for help

```
[1] win7.elf 14:05:46> yarascan?
file:            /home/scudette/rekall/rekall-core/rekall/plugins/windows/malware/yarascan.py
Plugin:          WinYaraScan (yarascan)
:                This is a Typed Plugin.
Positional Args:   pids: One or more pids of processes to select. (type: ArrayIntParser)
Keyword Args:
  hits:                     Quit after finding this many hits. (type: IntParser)
  string:                   A verbatim string to search for. (type: String)
  binary_string:            A binary string (encoded as hex) to search for. e.g. 000102[1-200]0506 (type: St
  yara_file:                The yara signature file to read. (type: String)
  yara_expression:          If provided we scan for this yara expression. (type: String)
  context:                  Context to print after the hit. (type: IntParser)
  pre_context:              Context to print before the hit. (type: IntParser)
  scan_kernel_paged_pool:   Scan the kernel paged pool. (type: Boolean)
  scan_kernel_nonpaged_pool: Scan the kernel non-paged pool. (type: Boolean)
  scan_kernel_code:         Scan the kernel image and loaded drivers. (type: Boolean)
  scan_kernel_session_pools: Scan session pools for all processes. (type: Boolean)
  limit:                    The length of data to search in each selected region. (type: IntParser)
  scan_physical:            Scan the physical address space only. (type: Boolean)
  scan_kernel:              Scan the entire kernel address space. (type: Boolean)
  scan_process_memory:      Scan all of process memory. Uses process selectors to narrow down selections. (t
  eprocess:                 Kernel addresses of eprocess structs. (type: ArrayIntParser)
  proc_regex:               A regex to select a process by name. (type: RegEx)
  method:                   Method to list processes. (type: ChoiceArray)
Docstring:       Scan using yara signatures.
Link:            http://www.rekall-forensic.com/epydocs/rekall.plugins.windows.malware.yarascan.WinYaraScan-c
```

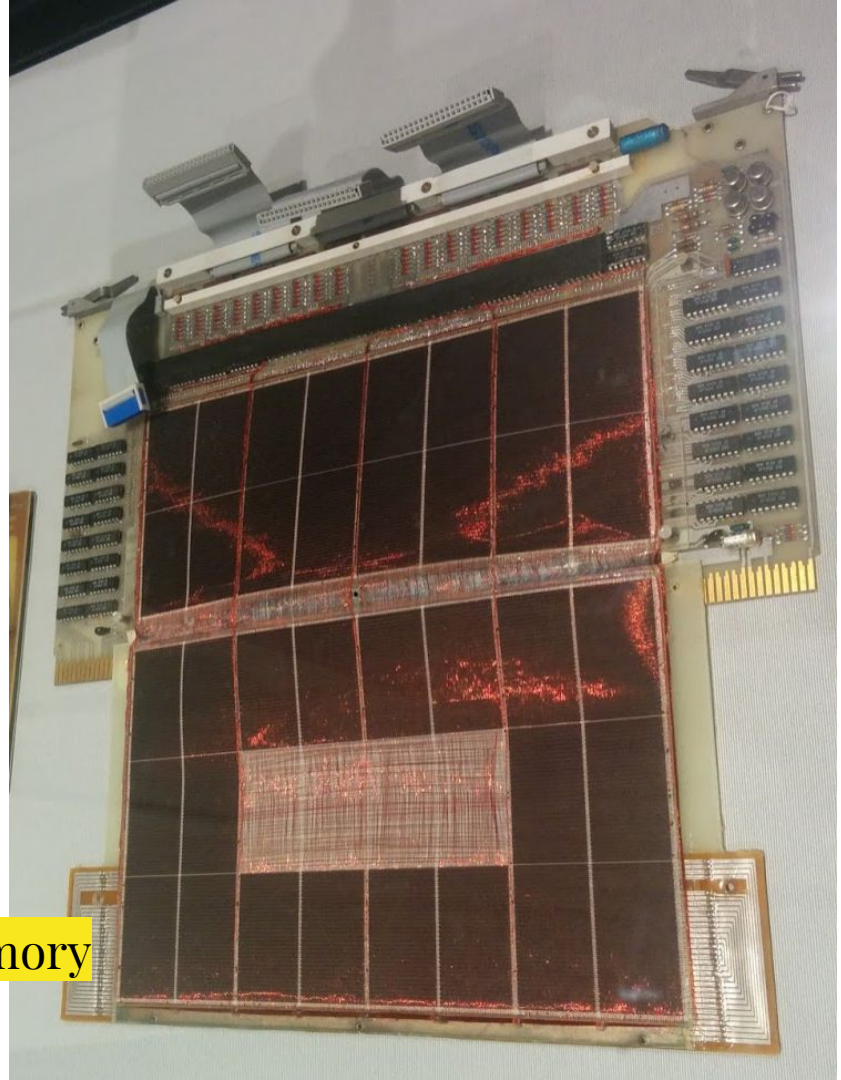
Rekall profiles

What is a profile?

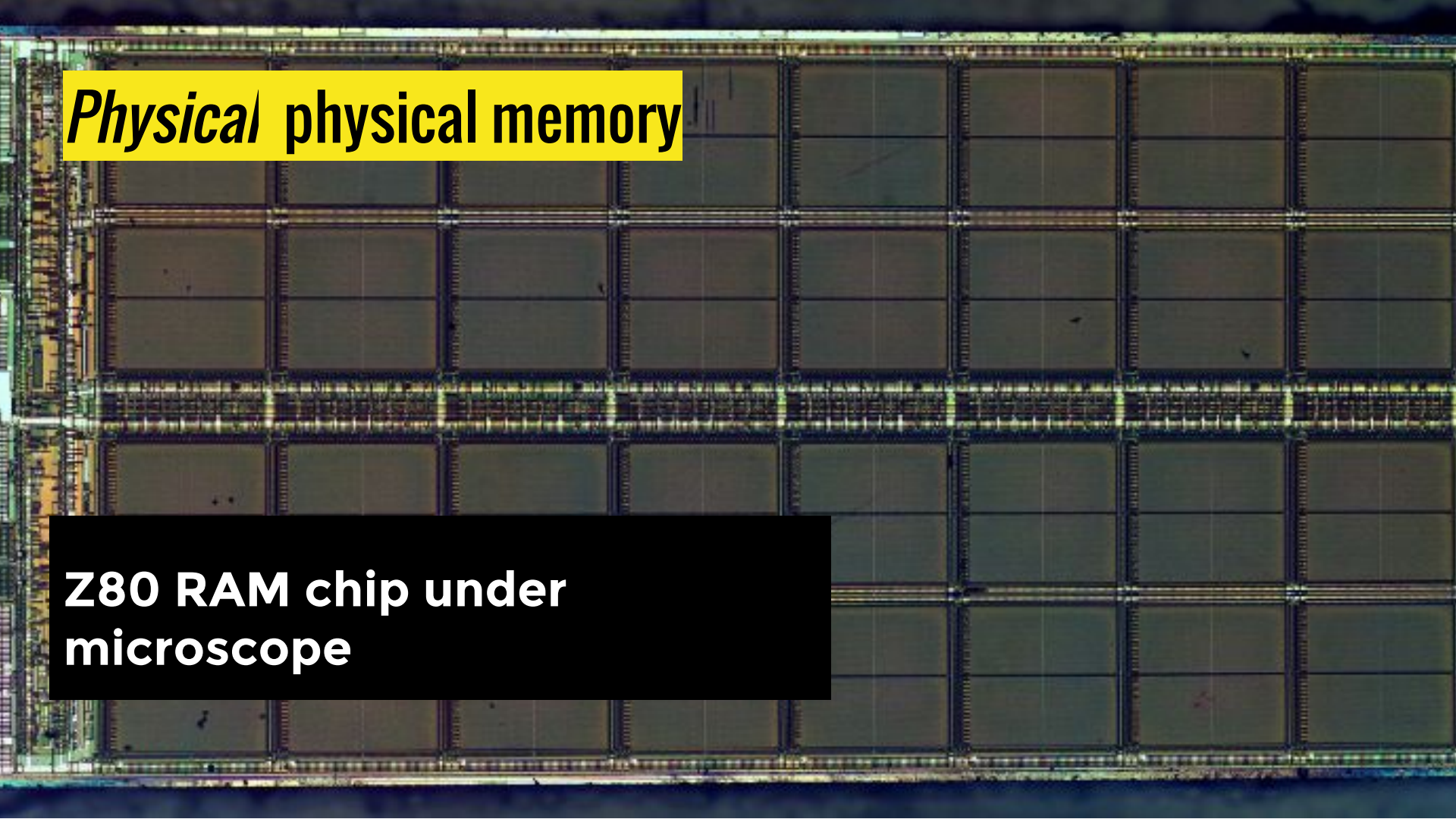
- OS specific information required for parsing the specific image involved.
  - Every OS version has a unique profile.
- Profiles live in the profile repository - stored as JSON data files.
  - Profile repository currently has
    - > 2000 windows profiles
    - > 600 Linux profiles
    - > 50 OSX profiles.
- Rekall autodetects the needed profile by itself.

Memory Analysis - Lightning introduction.

(1) What is memory and how does it work?

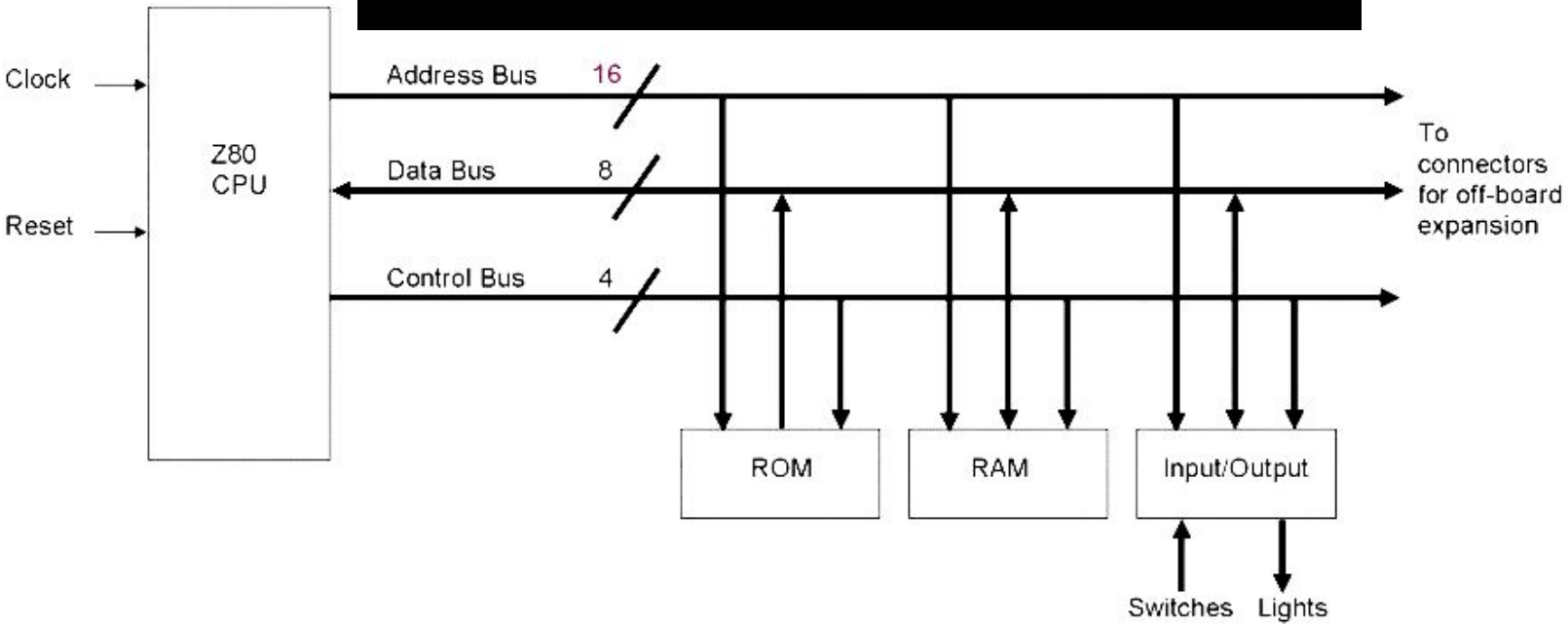


Handmade Core memory


Physical physical memory
Z80 RAM chip under microscope

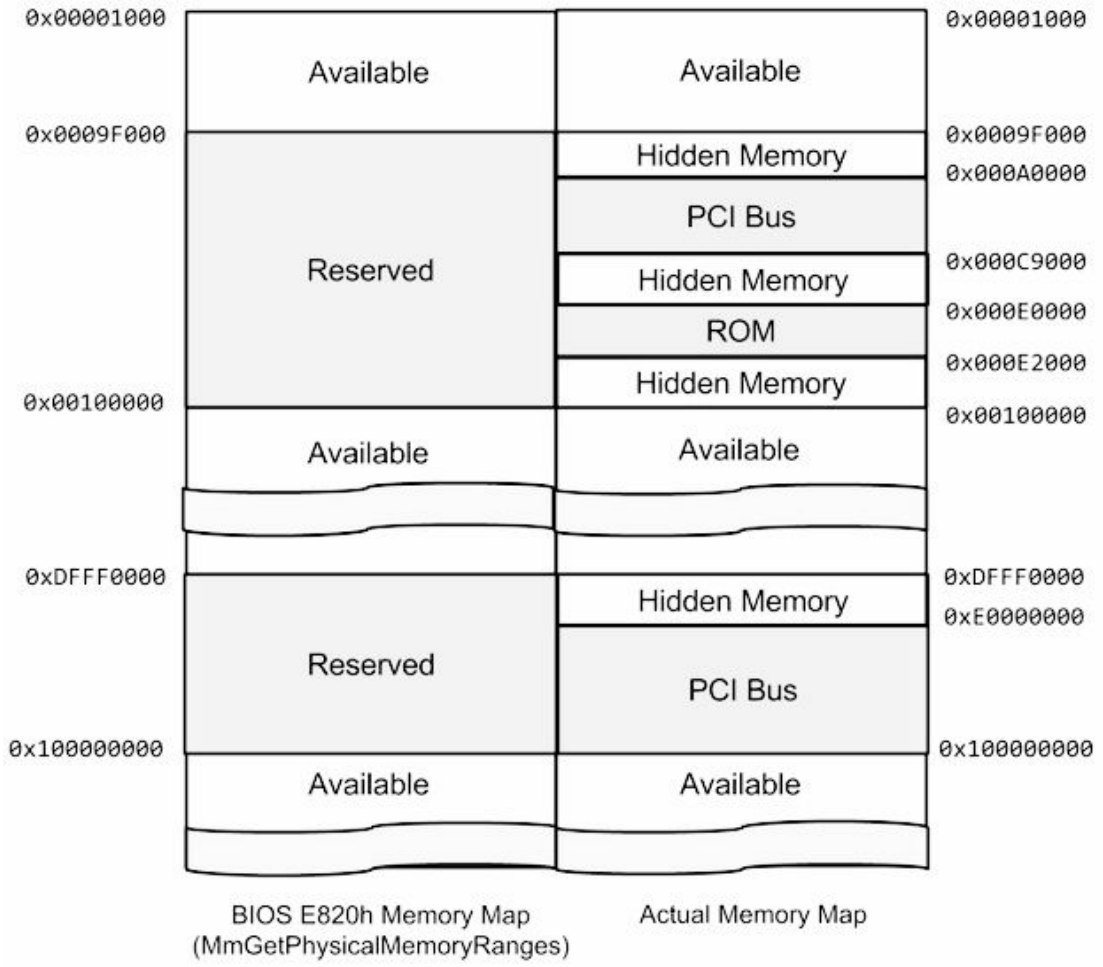
Logical view of memory

Computer Block Diagram



System Boot: Part I

1. The hardware sets up DMA
2. BIOS is loaded, and assigns addresses to devices
3. A Physical Memory Layout map is built and stored in RAM
4. (EFI) BootServices exits and leaves behind a copy
5. Registers are configured with high and low pointers to MMIO and DMA
6. Boot loader is started and the boot process ends

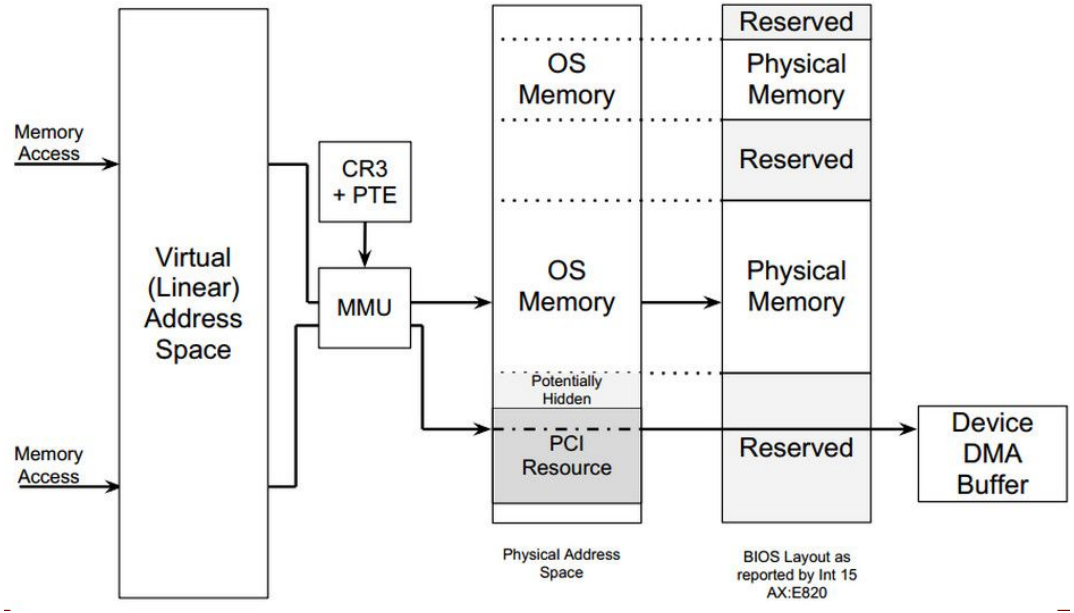


Reading reserved memory will blow up your computer.

(2) Virtual Memory

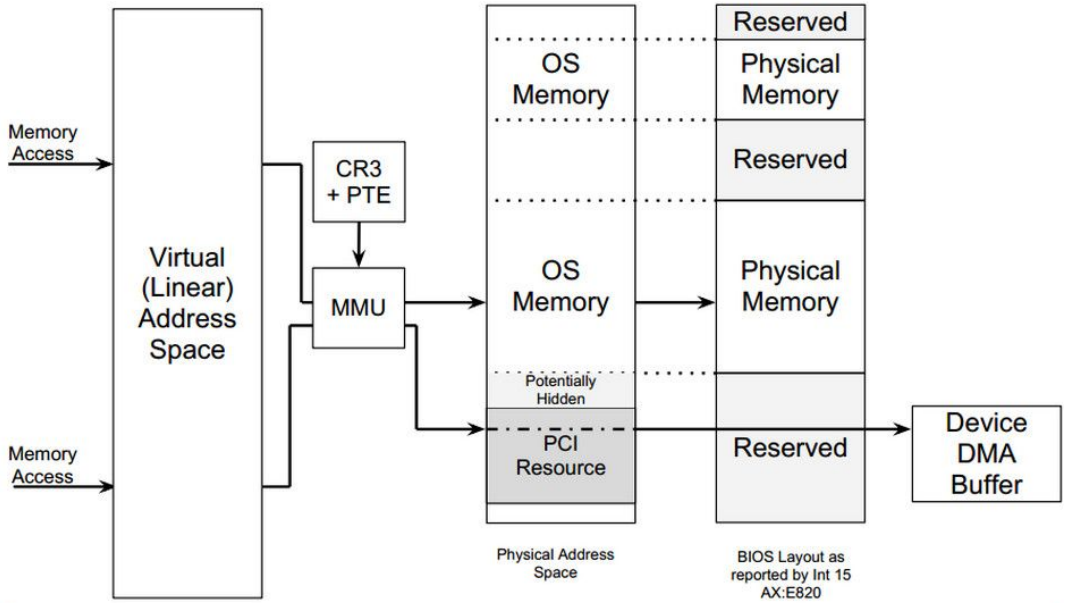
Boot Process: Part II

- The CPU starts in Real Mode
  - 16 bits
  - Addresses are real
  - 1 MiB addressable
- The kernel writes early page tables, setting up “identity mapping”
- CR3 value is set to address of (Page) Directory Table Base (DTB)
- Switch to protected mode



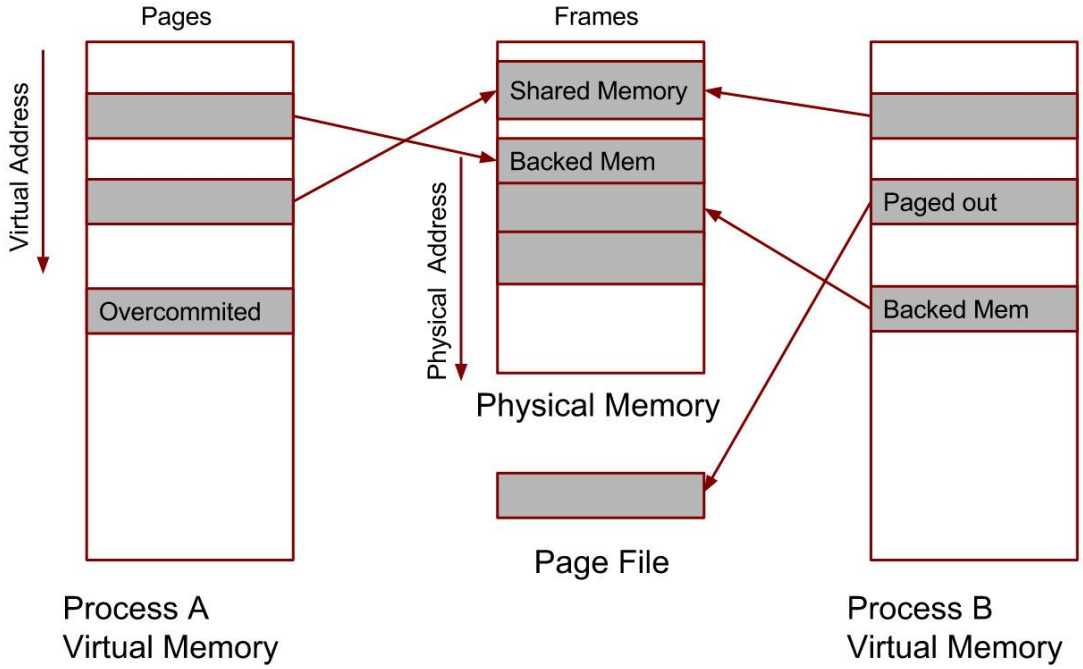
Boot Process: Part II cont.

- In protected mode:
  - 32 bits
  - Jumps are absolute
  - Segment registers work
  - 4 GiB addressable
  - Addresses are virtual
- On 64 bit, kernel further switches to long mode:
  - 64 bits (48 bit pointers)
  - Jumps are relative
  - Large (but not as large as you think) amounts of memory addressable



Virtual Memory

- Each process/task has a DTB pointer
- CR3 values are swapped on context switch
- Result: each process sees different memory
- Kernel (upper half) can see all and be seen by all



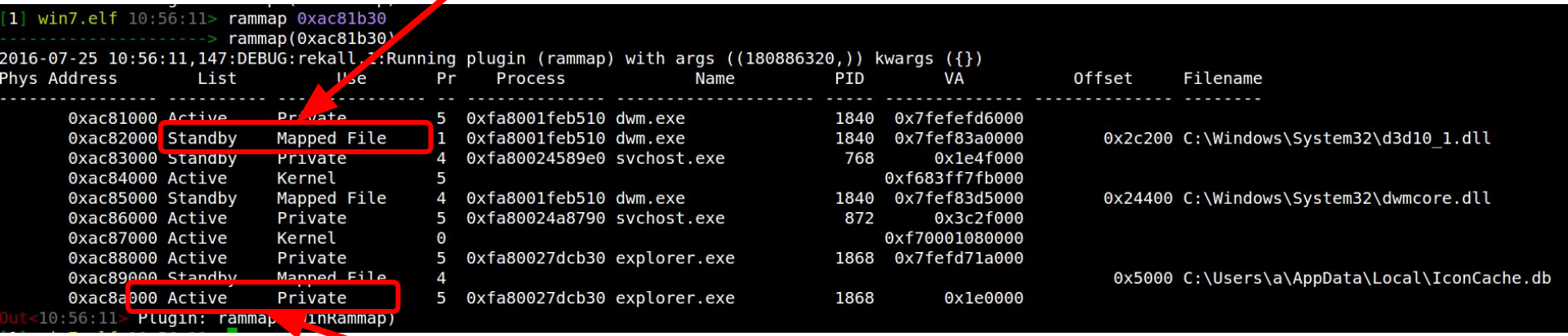
Navigating our way through memory

In the next exercise we navigate our way through memory to get a feel to how physical memory relates to virtual memory. There are some useful plugins:

- cc - Switch the process context to the process address space.
  - Makes the dis (disassembler), dump (hexdump a region of memory), address resolver etc - work with the specified process context.
- Rammap - just prints what each physical page is doing.
- vtop - Shows the Virtual to physical address translation process
- ptov - Translates the physical address to its virtual addresses.
- vad - Shows each process's memory regions and what they are used for.

The rammap plugin

File mapped into a process



Private memory for a process

The ptov plugin

File mapped into a process

This file is still mapped into memory but not in any process! This file is in the file cache.

```
[1] win7.elf 10:59:12> ptov 0xac82000
--------------------> ptov(0xac82000)
2016-07-25 10:59:12,738:DEBUG:rekall.1:Running plugin (ptov) with
File Mapping (C:\Windows\System32\d3d10 1.dll @ 0x2c200
Mapped in 0xfa8001feb510 dwm.exe  1840 @ 0x7fef83a0000
Out<10:59:12> Plugin: ptov (PtoV)
[1] win7.elf 10:59:17> ptov 0xac85000
--------------------> ptov(0xac85000)
2016-07-25 10:59:17,970:DEBUG:rekall.1:Running plugin (ptov) with
File Mapping (C:\Windows\System32\dwmcore.dll @ 0x24400
Mapped in 0xfa8001feb510 dwm.exe  1840 @ 0x7fef83d5000
Out<10:59:17> Plugin: ptov (PtoV)
[1] win7.elf 10:59:23> ptov 0xac89000
--------------------> ptov(0xac89000)
2016-07-25 10:59:23,684:DEBUG:rekall.1:Running plugin (ptov) with
File Mapping (C:\Users\a\AppData\Local\IconCache.db @ 0x5000
Out<10:59:23> Plugin: ptov (PtoV)
```

Example 2: Shared pages

Lets look at the lsass.exe process.

```
[1] win7.elf 11:05:29> pslist
--------------------> pslist()
2016-07-25 11:05:29,266:DEBUG:rekall.1:Running plugin (pslist) with
  _EPROCESS           Name           PID   PPID   Thds     Hnds    S
-------------- -------------------- ----- ------ ------ -------- --
0xfa80008959e0 System                  4      0     84      511
0xfa80024f85d0 svchost.exe           236    480     19      455
0xfa8001994310 smss.exe              272      4      2       29
0xfa8002259060 csrss.exe             348    340      9      436
0xfa8000901060 wininit.exe           384    340      3       75
0xfa8000900420 csrss.exe             396    376      8      192
0xfa8002282710 winlogon.exe          436    376      4      125
0xfa800206d5f0 services.exe          480    384     11      208
0xfa800183ab30 lsass.exe             496    384      8      707
0xfa800239db30 lsm.exe               504    384     10      143
0xfa80028a1640 WmiPrvSE.exe          592    608      9      176
0xfa80023f6770 svchost.exe           608    480     12      352
```

Switch context to lsass.exe

```
[1] win7.elf 11:06:17> cc 496
--------------------> cc(496)
2016-07-25 11:06:17,760:DEBUG:rekall.1:Running plugin (cc) with args ((496,)) kwargs ({})
2016-07-25 11:06:17,764:DEBUG:rekall.1:Switching to process context: lsass.exe (Pid 496@0xfa800183ab30)
Switching to process context: lsass.exe (Pid 496@0xfa800183ab30)
Out<11:06:17> Plugin: cc (WindowsSetProcessContext)
[1] win7.elf 11:06:25> vad 496
--------------------> vad(496)
2016-07-25 11:06:25,612:DEBUG:rekall.1:Running plugin (vad) with args ((496,)) kwargs ({})
-      VAD       lev   Start Addr      End Addr     com   ------- ------       Protect        Filename
  ------------- --- ------------- ------------- ------               ------------------- --------
------------------------------
0xfa800183ab30 lsass.exe   496
------------------------------
  0xfa80020b6e30   7        0x10000        0x1ffff      0 Mapped         READWRITE
  0xfa800280e680   6        0x20000        0x20fff      0 Mapped         READWRITE
  0xfa800239e010   7        0x30000        0x33fff      0 Mapped         READONLY
  0xfa800239ea20   5        0x40000        0x40fff      0 Mapped         READONLY
```

```
  0xfa80024a50d0   5      0x13d0000      0x14cffff    230 Private        READWRITE
  0xfa8002a9d650   6      0x16b0000      0x172ffff     15 Private        READWRITE
  0xfa80023c05e0   7     0x75140000     0x75141fff      0 Mapped  Exe    EXECUTE_WRITECOPY    \Windows\System32\msprivs.dll
  0xfa80022ae950   4     0x771f0000     0x7730efff      4 Mapped  Exe    EXECUTE_WRITECOPY    \Windows\System32\kernel32.dll
  0xfa80023bc600   6     0x77310000     0x77409fff      3 Mapped  Exe    EXECUTE_WRITECOPY    \Windows\System32\user32.dll
  0xfa800183aa50   5     0x77410000     0x775bafff     12 Mapped  Exe    EXECUTE_WRITECOPY    \Windows\System32\ntdll.dll
```

Find physical address for a virtual address

Rekall is showing all the steps in translating the virtual address to a physical address.

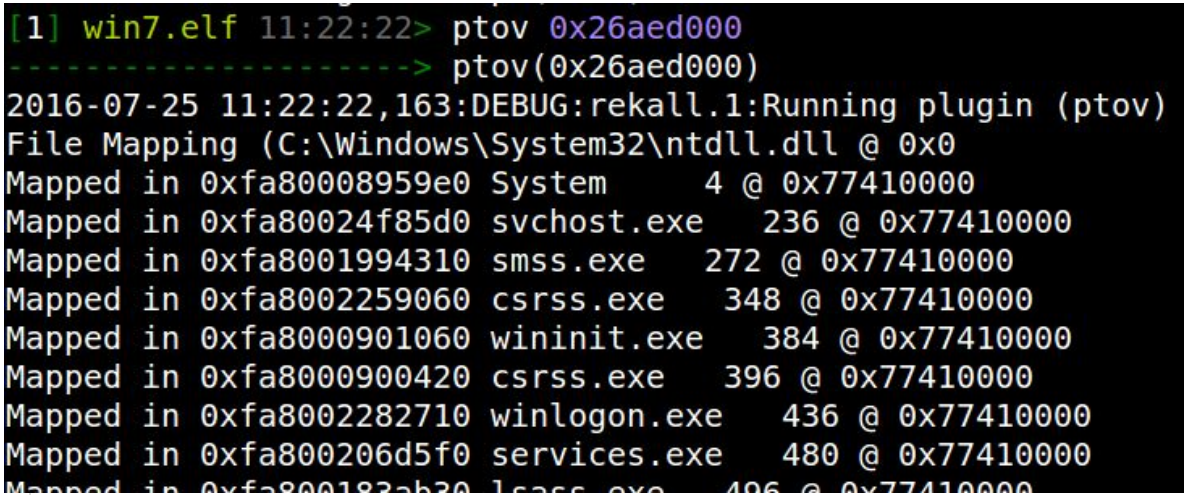
NOTE: This depends on the currently loaded process context.

```
[1] win7.elf 11:22:10> vtop 0x77410000
--------------------> vtop(0x77410000)
2016-07-25 11:22:10,276:DEBUG:rekall.1:Running plugin (vtop) with args ((2000748544,)) kw

****************** 0x77410000 ******************
Virtual 0x000077410000 Page Directory 0x17851000
pml4e@ 0x17851000 = 0x23e0000017792867
pdpte@ 0x17792008 = 0x18000001739d867
pde@ 0x1739ddd0 = 0x19000001735e867
pte@ 0x1735e080 = 0x8350000026aed005
[_MMPTE_HARDWARE Hard] @ 0x00001735e080
Offset             Field                Content
------ ------------------------------ -------
  0x0    Accessed                       [BitField(5-6):Accessed]: 0x00000000
  0x0    CacheDisable                   [BitField(4-5):CacheDisable]: 0x00000000
  0x0    CopyOnWrite                    [BitField(9-10):CopyOnWrite]: 0x00000000
  0x0    Dirty                          [BitField(6-7):Dirty]: 0x00000000
  0x0    Dirty1                         [BitField(1-2):Dirty1]: 0x00000000
  0x0    Global                         [BitField(8-9):Global]: 0x00000000
  0x0    LargePage                      [BitField(7-8):LargePage]: 0x00000000
  0x0    NoExecute                      [BitField(63-64):NoExecute]: 0x00000001
  0x0    Owner                          [BitField(2-3):Owner]: 0x00000001
  0x0    PageFrameNumber                [BitField(12-48):PageFrameNumber]: 0x00026AED
  0x0    SoftwareWsIndex                [BitField(52-63):SoftwareWsIndex]: 0x00000035
  0x0    Unused                         [BitField(10-11):Unused]: 0x00000000
  0x0    Valid                          [BitField(0-1):Valid]: 0x00000001
  0x0    Write                          [BitField(11-12):Write]: 0x00000000
  0x0    WriteThrough                   [BitField(3-4):WriteThrough]: 0x00000000
  0x0    reserved1                      [BitField(48-52):reserved1]: 0x00000000
Physical Address 0x26aed000
```

Find virtual address for a physical address

Note that the same physical page can be mapped in many processes **at the same time - shared memory**.

The file ntdll.dll is mapped in every process in windows. In theory it does not have to be mapped to the same address but it seems to be.



Questions

1. Can I dump files from memory?

    Yes, but ....

2. What happens when a dll is patched in memory?

    Page goes from shared to private (Copy on write).

3. My yarascan is showing an interesting string in physical memory. Who owns it?

Live analysis and Triaging

As memory sizes get bigger it is becoming more important to triage analysis before acquisition.

rekall --live

Insert memory drivers and gain access to raw physical memory.

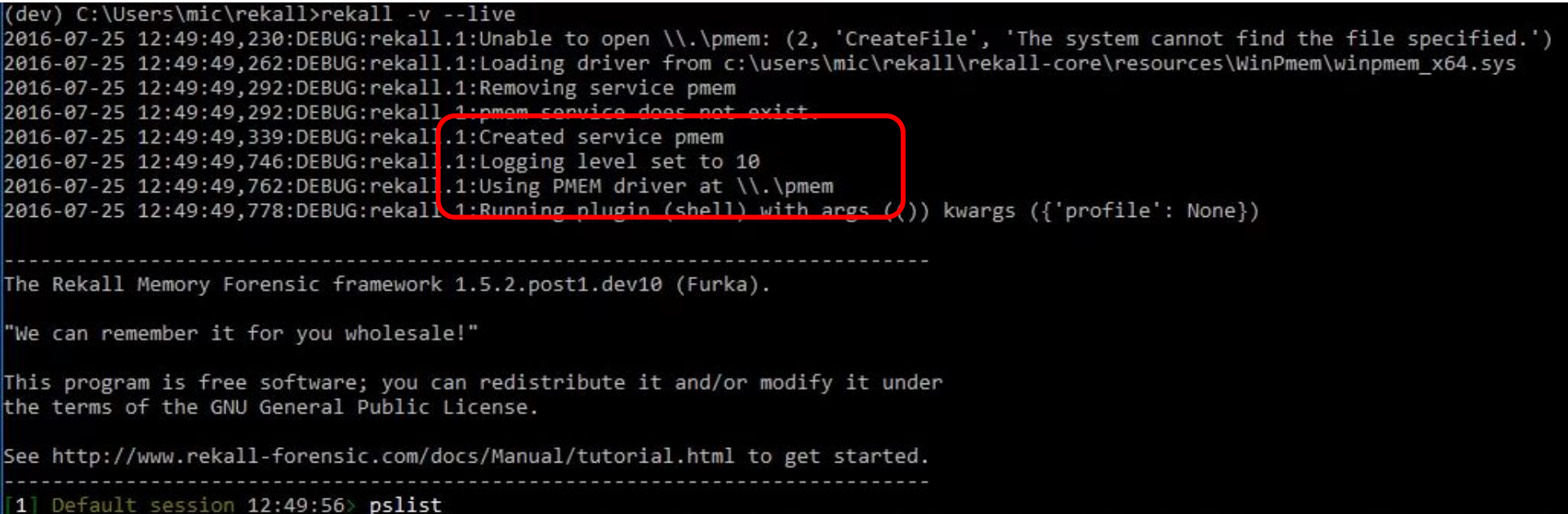
rekall --live API

Load the API analysis mode.
Analysis is done using the OS APIs.

rekall –live

Inserts driver to gain access to physical memory.

Operates against physical memory instead of image.



rekall –live API

Uses OS APIs to analyse memory.

```
(dev) C:\Users\mic\rekall>rekall --live API

----------------------------------------------------------------------------
The Rekall Memory Forensic framework 1.5.2.post1.dev10 (Furka).

"We can remember it for you wholesale!"

This program is free software; you can redistribute it and/or modify it under
the terms of the GNU General Public License.

See http://www.rekall-forensic.com/docs/Manual/tutorial.html to get started.
----------------------------------------------------------------------------
[1] Default session 12:57:55> pslist
----------------------------> pslist()
            Name                 pid    PPID   Thds    Hnds   Wow64          Start           binary
------------------------------ ------ ------ ------ -------- ------ ---------------------- ------
System Idle Process                 0      0      2          False  2016-07-25 19:37:08Z
System                              4      0    128          False  2016-07-25 19:37:08Z
smss.exe                          288      4      2          False  2016-07-25 19:37:07Z
VBoxService.exe                   348    572     11          False  2016-07-25 19:37:16Z
csrss.exe                         368    360     10          False  2016-07-25 19:37:13Z
wininit.exe                       444    360      3          False  2016-07-25 19:37:14Z
csrss.exe                         460    436     11          False  2016-07-25 19:37:14Z
winlogon.exe                      532    436      4          False  2016-07-25 19:37:14Z
services.exe                      572    444      4          False  2016-07-25 19:37:14Z
lsass.exe                         580    444     10          False  2016-07-25 19:37:14Z
```

API access Vs. Memory analysis

API Access:

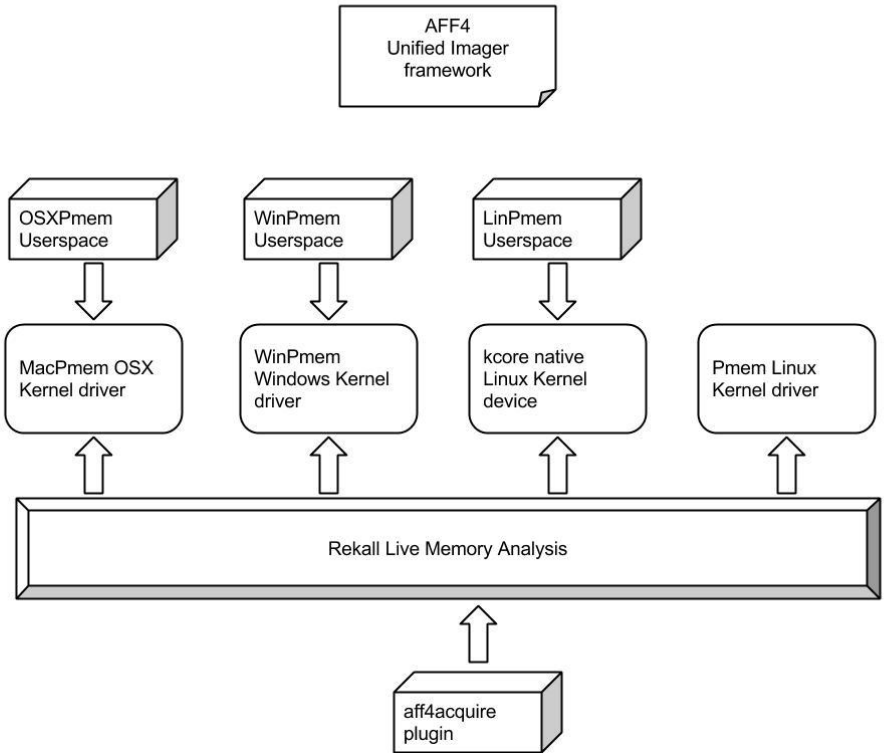
- Will always work.
- Does not need profiles.
- Very fast.
- Not very thorough - can only see what the OS APIs allow.
- More susceptible to malware manipulation.
- Fewer plugins available (Let's write more!)

Memory analysis

- May fail due to profile issues.
- Needs profiles with debugging info.
- Less fast.
- Can perform deep analysis (undocumented structs).
- Less susceptible to anti-forensics.

Memory acquisition

- After initial triage we would like to acquire memory.
  - Capture system state for later analysis.



Winpmem acquisition

- Winpmem is a C++ self contained acquisition tool.
  - Small footprint.
  - Minimal analysis.
- Usage examples:
  - AFF4 Zip file: **winpmem -o output.aff4 -c snappy**
  - AFF4 Directory: **winpmem -o output.dir\ -c snappy**
  - Raw, padded format in a zip file: **winpmem -o output.aff4 --format raw**
  - Raw, padded format in a directory: **winpmem -o output.dir/--format raw**

More information:

http://rekall-forensic.blogspot.ch/2016/05/the-pmem-suite-of-memory-acquisition.html

Acquisition through Rekall

- Rekall itself may be used to acquire memory.
  - This allows triaging and analysis to be done prior to acquisition.
  - Allows Artifact Collector to be used (more on that later).

Structured output

- Most Rekall plugins produce tabular output:

```
[1] win7.elf 13:59:28> pslist
----------------------> pslist()
2016-07-25 13:59:28,671:DEBUG:rekall.1:Running plugin (pslist) with args (()) kwargs ({})
  _EPROCESS           Name          PID   PPID   Thds    Hnds    Sess  Wow64          Start                       Exit
-------------- -------------------- ----- ------ ------ -------- ------ ------ ------------------------ ------------------------
0xfa80008959e0 System                  4      0     84      511      - False  2012-10-01 21:39:51Z     -
0xfa80024f85d0 svchost.exe           236    480     19      455      0 False  2012-10-01 14:40:01Z     -
0xfa8001994310 smss.exe              272      4      2       29      - False  2012-10-01 21:39:51Z     -
0xfa8002259060 csrss.exe             348    340      9      436      0 False  2012-10-01 21:39:57Z     -
0xfa8000901060 wininit.exe           384    340      3       75      0 False  2012-10-01 21:39:57Z     -
0xfa8000900420 csrss.exe             396    376      8      192      1 False  2012-10-01 21:39:57Z     -
```

What if we wanted to customize the output?

Structured output

- We can describe each plugin output

```
[1] win7.elf 14:09:04> describe pslist
----------------------> describe(pslist)
2016-07-25 14:09:04,988:DEBUG:rekall.1:Running plugin (describe) with
                  Field                                  Type
------------------------------------------------------- ----
  _EPROCESS                                             _EPROCESS
  ppid                                                  unsigned int
  thread_count                                          unsigned long
  handle_count                                          unsigned int
  session_id                                            unsigned int
  wow64                                                 bool
  process_create_time                                   WinFileTime
  process_exit_time                                     WinFileTime
```

Efilter queries

Efilter queries are SQL like but the plugins are executed immediately

```
1] win7.elf 14:13:36> select * from pslist() where ppid==2616
2016-07-25 14:13:36,339:DEBUG:rekall.1:Running plugin (search) with args (()) kwargs ({'query': u'select
  _EPROCESS           Name          PID   PPID   Thds    Hnds    Sess  Wow64          Start
------------ ------------------ ----- ------ ------ -------- ------ ------ ----------------------
0xfa8002ad0190 cmd.exe              2644   2616      2       66      1 True   2012-10-01 14:40:20Z
```

```
[1] win7.elf 14:23:15> select _EPROCESS.name, _EPROCESS.pid, _EPROCESS.ActiveThreads
                  ...:  from pslist() order by _EPROCESS.ActiveThreads desc limit 5
    name       pid  ActiveThreads
------------ ----- -------------

System         4     84
svchost.exe  932     44
svchost.exe  872     30
svchost.exe  768     23
svchost.exe  1192    21
```

Rekall response plugins

When operating in live mode it is possible to execute many incident response plugins through Rekall:

glob - search for files using a glob expression.

file_yara - Apply yara signatures on files.

wmi - Run WMI queries

and many more.

Run a vara rule against all executables in the windows directory.


```
rule r1 {
  strings:
      $a = "Microsoft" wide
  condition:
      any of them
}
```

Yara rule

```
select path.filename from
glob("c:\windows\*.exe")
```

Path glob

```
select * from file_yara(
  paths: (select path.filename from
          glob("c:\windows\*.exe")).filename,
  yara_expression: "rule r1 {strings: $a =
\"Microsoft\" wide condition: any of them}")
```

Glob for all *.exe files in the windows directory and vara scan them for "Microsoft" encoded in UTF16.

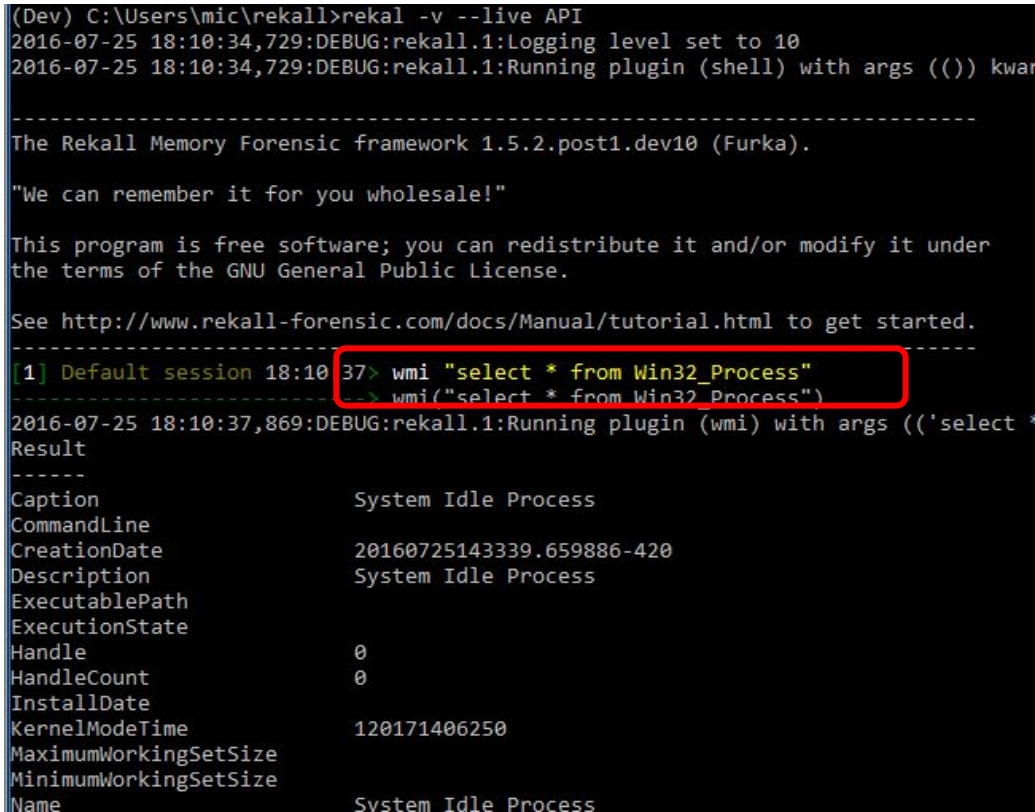

```
glob "c:\windows\*.exe"
```

Now use the search plugin to insert query parameters:

```
plugins.search('select * from
file_yara(paths: (select
path.filename from
glob("c:\windows\*.exe")).filename,
binary_string: {str})',
query_parameters=dict(str="Microsoft"
.encode("utf-16-le").encode("hex")))
```

Example - process listing through WMI

- WMI plugin allows arbitrary WMI queries to be issued.
- There are many resources for good IR WMI queries to issue.
- The wmi plugin allows to specify a different base object



Excercise - Use WMI

1. List local user accounts.
2. List installed hotfixes.
3. Which AV product is installed?
4. What are the MAC addresses? What IP addresses are currently assigned?

WMI is a very powerful IR tool!

Forensic Artifacts

- WMI is a powerful tool, but....
  - You need to know what to query for.
  - There are so many different providers, so much information.
- Artifacts are a way to package an expert's knowledge so it can easily be reused.
- YAML based definition language giving a name and description to a query.

```
name: WMIHotFixes
doc: |
   Installed hotfixes via Windows Management
   Instrumentation (WMI).
sources:
- type: WMI
  attributes:
    query: |
      SELECT * from Win32_QuickFixEngineering
labels:
  - Software
supported_os:
  - Windows
```

https://github.com/ForensicArtifacts/artifacts

Forensic Artifacts

- Artifacts maintained in a public repository.
- Rekall automatically checks latest artifacts definitions from profile repository.
- Types of artifacts available:
  - FILE - Collect files
  - REGISTRY_KEY/REGISTRY_VALUE
  - WMI

Available in Live Mode

Also available with Images

  - REKALL_EFILTER - Run Rekall plugin and collect output.
  - ARTIFACT_GROUPS - A way to group similar artifacts.

```
[1] win7.elf 10:16:09> artifact_list
--------------------> artifact_list()
2016-07-26 10:16:09,019:DEBUG:rekall.1:Running plugin (artifact_list) with args (()) kwargs ({})
            Name                OS          Labels              Types                            Description
------------------------------ -------- ------------------ -------------------- ----------------------------------------------
ProcessListing                 Windows,                    REKALL_EFILTER       A list of all processes from memory.
                               Darwin,
                               Linux
WindowsKernelModulesLoadPath   Windows                     REKALL_EFILTER       On windows, kernel modules should be loaded from
                                                                                the Windows directory. This
                                                                                artifact locates modules which are not loaded from
                                                                                this location. Such kernel
                                                                                modules might be suspicious because they are
                                                                                loaded from temporary paths or
                                                                                dropped by first stage loaders.
WindowsDriverIRPKeyLogger      Windows                     REKALL_EFILTER       One technique of installing a keylogger in windows
                                                                                is to hook the IRP handler
                                                                                routine of the kbd driver. This artifact dumps the
                                                                                entire IRP handler table
                                                                                for the kbdclass (Keyboard Class) driver, if any
                                                                                of the functions do not
                                                                                reside within the driver itself (i.e. it is
                                                                                hooked).
WindowsFakeSVCHost             Windows                     REKALL_EFILTER       Windows has several different svchost.exe
                                                                                processes normally. Sometimes
                                                                                malware names itself as svchost.exe to confuse
                                                                                curious users. However, the
                                                                                real svchost.exe are always launched from
                                                                                services.exe.
                                                                                This artifact finds those svchost.exe processes
                                                                                which are not launches by
                                                                                services.exe. This should not normally happen and
                                                                                it is suspicious if it does.
```

Forensic Artifacts

REKALL EFILTER artifacts

```
name: WindowsKernelModulesLoadPath
doc: |
  On windows, kernel modules should be loaded from the Windows directory. This
  artifact locates modules which are not loaded from this location. Such
  kernel modules might be suspicious because they are loaded from temporary
  paths or dropped by first stage loaders.

sources:
  - type: REKALL_EFILTER
    attributes:
      query: >
        select offset_v, file_name, module_base, module_size, path from
          modules() where not (path =~ "(?i)Windows")
      image_type:
        - Windows
      type_name: modules
      fields:
        - name: offset_v
          type: int
          style: address
        - name: file_name
          type: unicode
        - name: module_base
          type: int
          style: address
        - name: module_size
          type: int
          style: address
        - name: path
          type: unicode
```

Collecting Artifacts

```
[1] win7.elf 10:24:22> artifact_collector ["WindowsKernelModulesLoadPath", "WindowsDriverIRPKeyLogger"]
---------------------> artifact_collector(["WindowsKernelModulesLoadPath", "WindowsDriverIRPKeyLogger"])
2016-07-26 10:24:22,956:DEBUG:rekall.1:Running plugin (artifact_collector) with args ((['WindowsKernelMo
 ({})
--------------------------------------
Artifact: WindowsKernelModulesLoadPath
--------------------------------------
  offset_v       file_name   module_base    module_size    path
  --------       ---------   -----------    -----------    ----
  0xfa80028a4950 winDB9E.tmp 0xf88002ba7000         0xa000 C:\Users\a\AppData\Local\Temp\winDB9E.tmp
----------------------------------
Artifact: WindowsDriverIRPKeyLogger
----------------------------------
```

Collecting Artifacts - ARTIFACT_GROUPS

```
[1] Live (API) 14:08:21> artifact_collector "BrowserHistory"
-----------------------> artifact_collector("BrowserHistory")
2016-07-26 14:08:21,529:DEBUG:rekall.1:Running plugin (artifact_collector) with args (('BrowserHistory',)) kwargs ({})
----------------------
Artifact: BrowserHistory
----------------------
----------------------
Artifact: ChromeHistory
----------------------
  st_mode    st_nlink st_uid           st_gid     st_size  st_mtime      filename
  -------    -------- ------           ------     -------  --------      --------
  -rw-r----- 1        scudette (116417) eng (5000) 14024704 1469567034.65 /home/scudette/.config/google-chrome/Default/History
----------------------
Artifact: FirefoxHistory
----------------------
  st_mode    st_nlink st_uid           st_gid     st_size  st_mtime      filename
  -------    -------- ------           ------     -------  --------      --------
  -rw-r----- 1        scudette (116417) eng (5000) 10485760 1464825434.92 /home/scudette/.mozilla/firefox/w2ic50qy.default/places
2016-07-26 14:08:22,245:DEBUG:rekall.1:Skipping artifact InternetExplorerHistory: Supported OS: [u'Windows'], but we are Linux
----------------------
Artifact: OperaHistory
----------------------
2016-07-26 14:08:22,497:DEBUG:rekall.1:Skipping artifact SafariHistory: Supported OS: [u'Windows', u'Darwin'], but we are Linux
Out<14:08:22> Plugin: artifact_collector (ArtifactsCollector)
```

Collecting Artifacts - Collecting files

```
[1] Live (API) 14:23:08> artifact_collector "BrowserHistory", copy_files=True, output_path="/tmp/output.zip"
----------------------> artifact_collector("BrowserHistory", copy_files=True, output_path="/tmp/output.zip")
2016-07-26 14:23:08,092:DEBUG:rekall.1:Running plugin (artifact_collector) with args (('BrowserHistory',)) kwa
utput.zip'})
-----------------------
Artifact: BrowserHistory
-----------------------
-----------------------
Artifact: ChromeHistory
-----------------------
  st_mode    st_nlink st_uid           st_gid     st_size  st_mtime      filename
  -------    -------- ------           ------     -------  --------      --------
  -rw-r----- 1        scudette (116417) eng (5000) 14024704 1469568165.14 /home/scudette/.config/google-chrome
-----------------------
Artifact: FirefoxHistory
-----------------------
  st_mode    st_nlink st_uid           st_gid     st_size  st_mtime      filename
  -------    -------- ------           ------     -------  --------      --------
  -rw-r----- 1        scudette (116417) eng (5000) 10485760 1464825434.92 /home/scudette/.mozilla/firefox/w2ic
2016-07-26 14:23:09,871:DEBUG:rekall.1:Skipping artifact InternetExplorerHistory: Supported OS: [u'Windows'],
----------------------
Artifact: OperaHistory
----------------------
2016-07-26 14:23:10,122:DEBUG:rekall.1:Skipping artifact SafariHistory: Supported OS: [u'Windows', u'Darwin'],
Out<14:23:10> Plugin: artifact_collector (ArtifactsCollector)
[1] Live (API) 14:23:21> !unzip -l /tmp/output.zip
Archive:  /tmp/output.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
 14024704  2016-07-26 14:22   home/scudette/.config/google-chrome/Default/History
      582  2016-07-26 14:23   artifacts/ChromeHistory.json
      181  2016-07-26 14:23   artifacts/ChromeHistory.csv
 10485760  2016-06-01 16:57   home/scudette/.mozilla/firefox/w2ic50qy.default/places.sqlite
      594  2016-07-26 14:23   artifacts/FirefoxHistory.json
      192  2016-07-26 14:23   artifacts/FirefoxHistory.csv
      366  2016-07-26 14:23   artifacts/OperaHistory.json
       58  2016-07-26 14:23   artifacts/OperaHistory.csv
---------                     -------
 24512437                     8 files
```

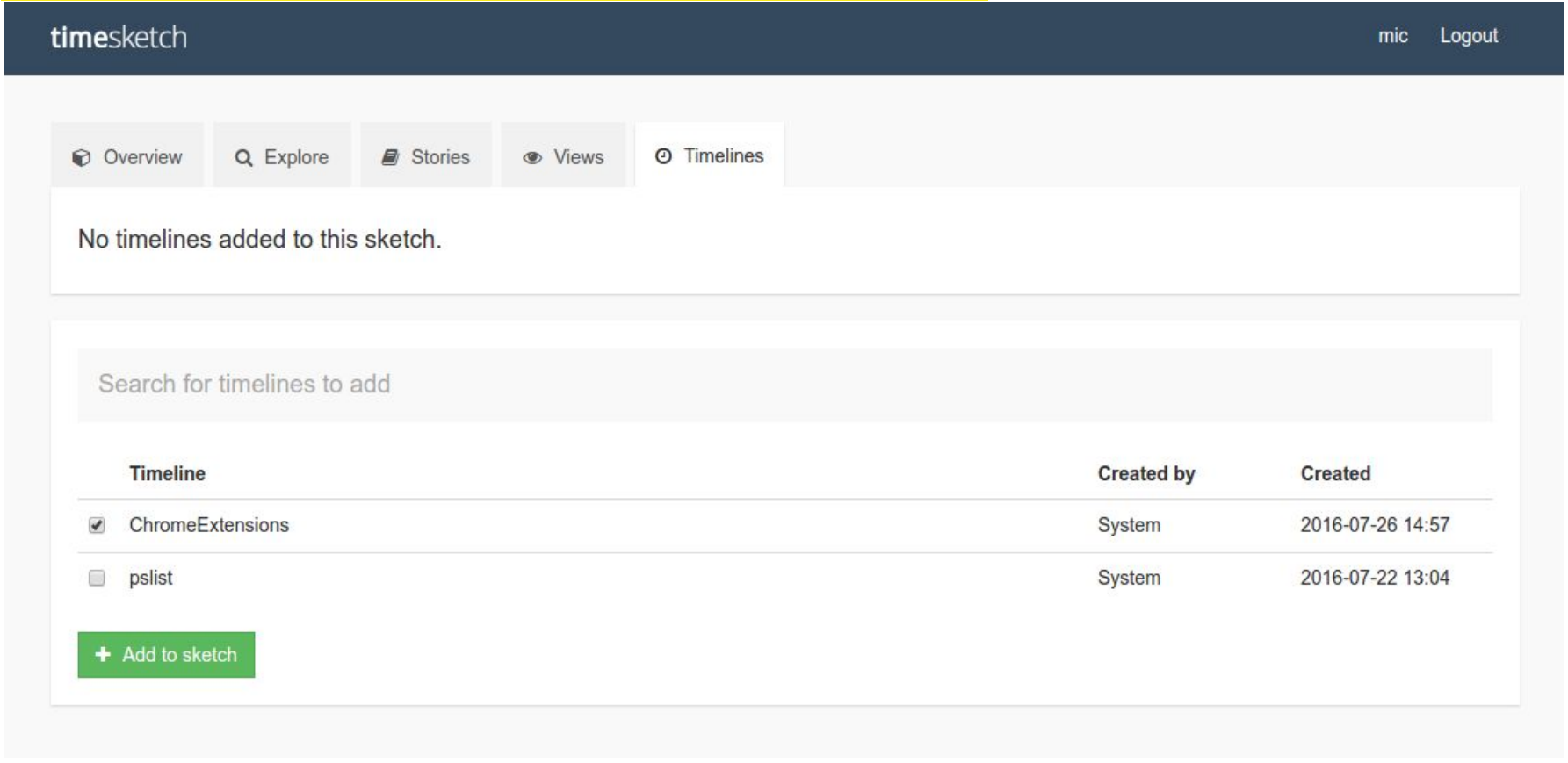
Collecting Artifacts - Collecting timelines

```
[1] Live (API) 14:54:36> artifact_collector ["KernelModules", "ChromeExtensions"], create_timeline=True, output_path="/tmp/output.zip"
----------------------> artifact_collector(["KernelModules", "ChromeExtensions"], create_timeline=True, output_path="/tmp/output.zip")
2016-07-26 14:54:36,291:DEBUG:rekall.1:Running plugin (artifact_collector) with args ((['KernelModules', 'ChromeExtensions'],)) kwargs ({'create_timeline': Tr
ue, 'output_path': '/tmp/output.zip'})
----------------------
Artifact: KernelModules
----------------------
  st_mode    st_nlink st_uid   st_gid   st_size st_mtime      filename
  -------    -------- ------   ------   ------- --------      --------
  -rw-r--r-- 1        root (0) root (0) 30      1337328217.0  /etc/modprobe.d/vmwgfx-fbdev.conf
  -rw-r--r-- 1        root (0) root (0) 583     1300471867.0  /etc/modprobe.d/blacklist-rare-network.conf
  -rw-r--r-- 1        root (0) root (0) 210     1300471867.0  /etc/modprobe.d/blacklist-firewire.conf
  -rw-r--r-- 1        root (0) root (0) 237     1415055846.26 /etc/modprobe.d/broadcom-btusb.conf
```

Load the timeline into Timesketch https://github.com/google/timesketch

```
(Dev) scudette@scudette-glaptop:/tmp$ tsctl csv2ts --name ChromeExtensions --file /tmp/artifacts/ChromeExtensions.timeline.csv --index_name test
Events inserted: 1000
Events inserted: 2000
Events inserted: 3000
Timeline name: ChromeExtensions
Elasticsearch index: test
```

Timesketch - a timeline analysis tool



timesketch

mic Logout

Overview | Explore | Stories | Views | Timelines

*manifest.json

Filters | Charts | Starred | Save view

45 events (0.036s)

Sort | Export | Toggle all | Add star | Remove star

2014-06-13T09:21:43+00:00	[ChromeExtensions] -rw------- 1 scudette (116417) eng (5000) 656 1402651303.75 /home/scudette/.config/google-chrome/Default/Extensions/lgkfcdkljfpbaibkeplffhlbifmkckel/0.1_0/manifest.json	ChromeExtensions
2014-06-13T09:21:43+00:00	[ChromeExtensions] -rw------- 1 scudette (116417) eng (5000) 1012 1402651303.57 /home/scudette/.config/google-chrome/Default/Extensions/aagapegjmpgahkpldpjbnjpjbngnhcjl/1.4_0/manifest.json	ChromeExtensions

datetime	2014-06-13T09:21:43+00:00
filename	/home/scudette/.config/google-chrome/Default/Extensions/aagapegjmpgahkpldpjbnjpjbngnhcjl/1.4_0/manifest.json
message	-rw------- 1 scudette (116417) eng (5000) 1012 1402651303.57 /home/scudette/.config/google-chrome/Default/Extensions/aagapegjmpgahkpldpjbnjpjbngnhcjl/1.4_0/manifest.json
st_gid	eng (5000)
st_mode	-rw-------
st_mtime	1402651303.574779
st_nlink	1
st_size	1012
st_uid	scudette (116417)
timestamp	1402651303

Mic

Malicious extension.

Tue, 26 Jul 2016 15:12:12 -0000

What's on your mind?

Post comment | Cancel

GRR Response Rig

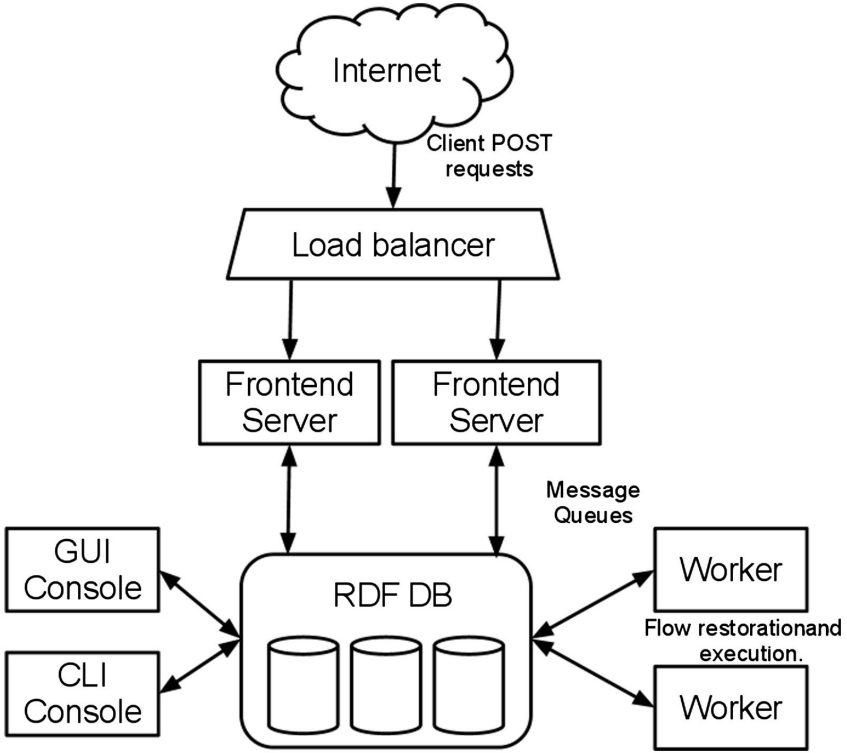
What is GRR

- GRR Is an Agent based incident response framework.
- Agents deployed in advance
- Agents poll the server for jobs
- Server schedules
  - Jobs for specific clients.
  - Hunts to search for indicators on all clients.

GRR
RAPID RESPONSE

Architecture

- Client
  - grr_client + client components
- Frontend Server
  - grr_server --component http_server
- Admin UI
  - grr_server --component ui
- Worker
  - grr_server --component worker
- Console
  - grr_console



Workshop

- In this workshop we will learn about GRR internals.
- We will install from source and build our own clients.
- We run all components by hand and watch debugging messages.
- Try to understand how the different components fit together.
- In real life you would install on an Ubuntu server with proper systemd support - logs go to syslog.
- For this workshop we assume that we do not have internet connectivity
    - All software is distributed in the wheelhouse.

Install GRR

- Create a new Virtual Env.

```
c:\python27\scripts\virtualenv e:\GRRDev
```

- Activate it

```
e:\GRRDev\Scripts\activate
```

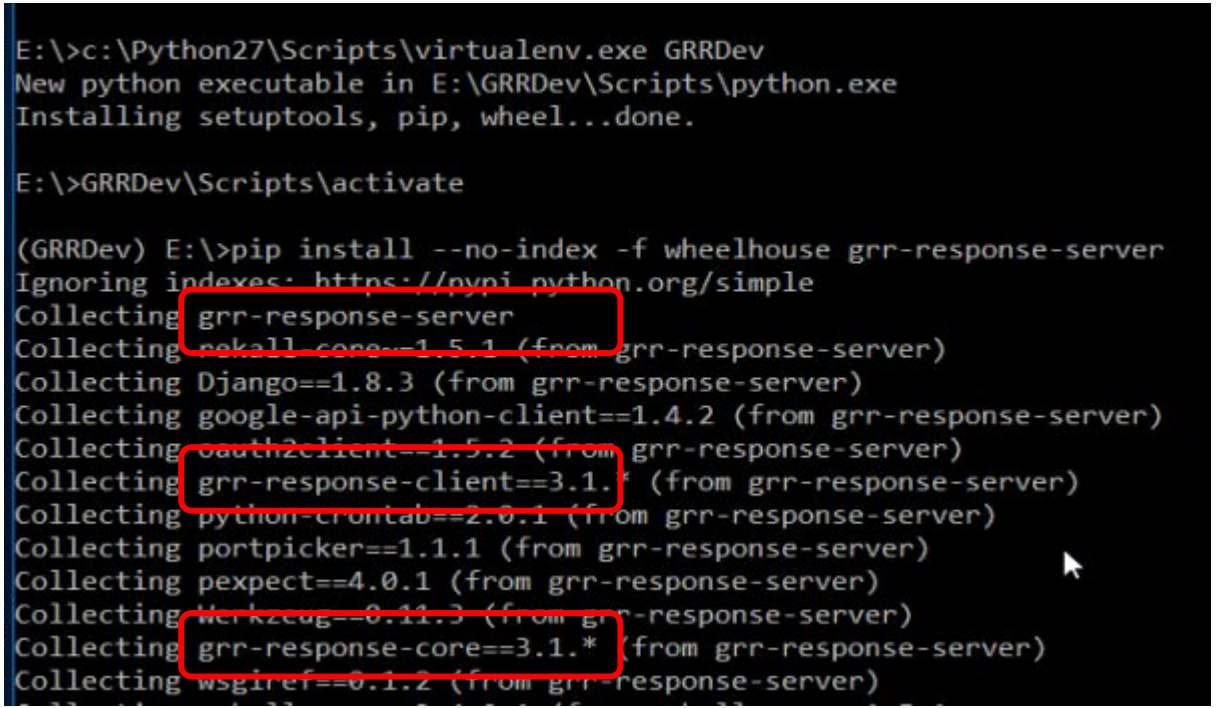
- Install the core of GRR

```
set PIP_FIND_LINKS=e:\wheelhouse

set PIP_NO_INDEX=1

pip install grr-response-server
```

Please specify this to make PIP use the local wheelhouse directory. Wifi is really slow and we don't all want to wait.

Please specify this to make PIP ignore the public pypi index.

```
E:\>c:\Python27\Scripts\virtualenv.exe e:\dev
New python executable in e:\dev\Scripts\python.exe
Installing setuptools, pip, wheel...done.

E:\>e:\dev\Scripts\activate

(dev) E:\>set PIP_FIND_LINKS=e:\wheelhouse

(dev) E:\>set PIP_NO_INDEX=y

(dev) E:\>pip install grr-response-server
Ignoring indexes: https://pypi.python.org/simple
Collecting grr-response-server
Collecting Django==1.8.3 (from grr-response-server)
```


E:\>c:\Python27\Scripts\virtualenv.exe GRRDev
New python executable in E:\GRRDev\Scripts\python.exe
Installing setuptools, pip, wheel...done.
E:\>GRRDev\Scripts\activate
(GRRDev) E:\>pip install --no-index -f wheelhouse grr-response-server
Ignoring indexes: https://pypi.python.org/simple
Collecting grr-response-server
Collecting Django==1.8.3 (from grr-response-server)
Collecting google-api-python-client==1.4.2 (from grr-response-server)
Collecting grr-response-client==3.1.* (from grr-response-server)
Collecting portpicker==1.1.1 (from grr-response-server)
Collecting pexpect==4.0.1 (from grr-response-server)
Collecting grr-response-core==3.1.* (from grr-response-server)

Try Out

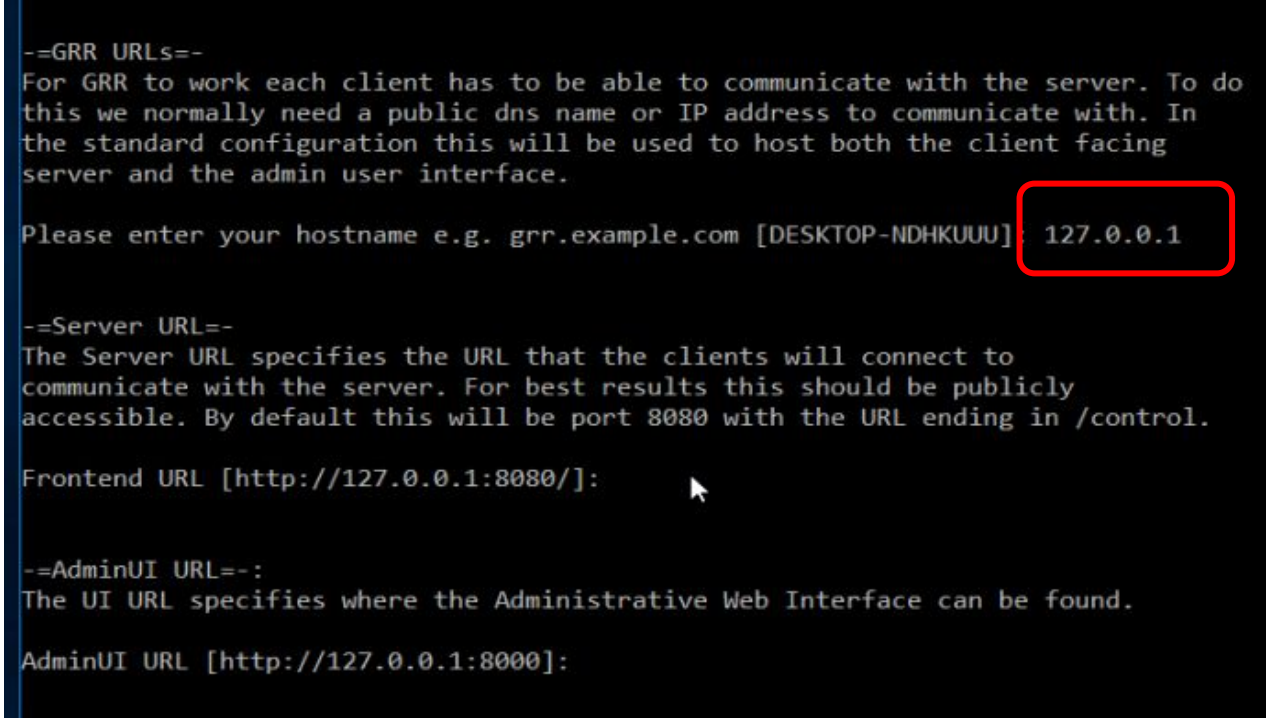
Initialize the GRR installtion.

- When installing GRR you need
  - To generate your own keys.
  - Configure your own data store
  - Point GRR clients at your public front end address.
  - Build customized clients with your own configuration.
    - GRR comes with many templates for multiple OS's and versions.
    - **A template** is basically a binary without a configuration.
    - **Repacking** the template means to inject the configuration and produce an installable package.

First thing to do: Initialize.

```
(GRRDev) E:\>grr_config_updater initialize
Checking write access on config e:\grrdev\install_data/etc/server.local.yaml

Step 0: Importing Configuration from previous installation.
No old config file found.

Step 1: Key Generation
All keys will have a bit length of 2048.
Generating executable signing key
Generating CA keys
Generating Server keys
Generating Django Secret key (used for xsrf protection etc)

Step 2: Setting Basic Configuration Parameters
We are now going to configure the server using a bunch of questions.


--=GRR Datastore=-
For GRR to work each GRR server has to be able to communicate with the
datastore.  To do this we need to configure a datastore.


1. SQLite (Default) - This datastore is stored on the local file system. If you
configure GRR to run as non-root be sure to allow that user access to the files.

2. MySQL - This datastore uses MySQL and requires MySQL 5.6 server or later
to be running and a user with the ability to create the GRR database and tables.
The MySQL client binaries are required for use with the MySQLdb python module as
well.

Datastore [1]:
Datastore Location [e:\grrdev\lib\site-packages\grr/var/grr-datastore]:
```

Configure GRR



We will just connect locally - be sure to set hostname to 127.0.0.1

Client Templates

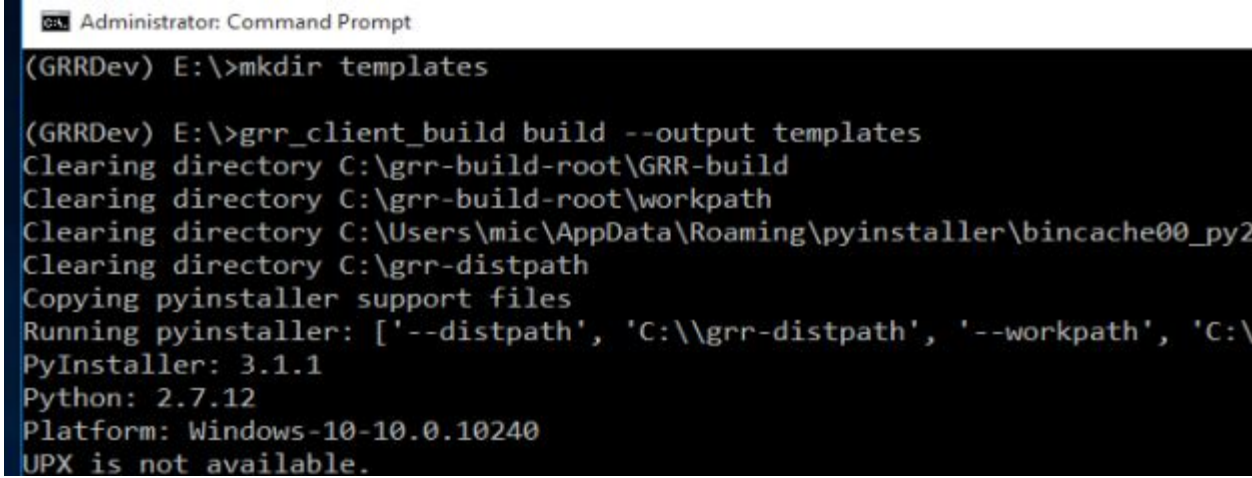
```
Step 4: Installing template package and repackaging clients with new configuration.
Download client templates? You can skip this if templates are already installed.[Yn]: n
Repack client templates?[Yn]: n

Initialization complete, writing configuration.
Please restart the service for it to take effect.
```

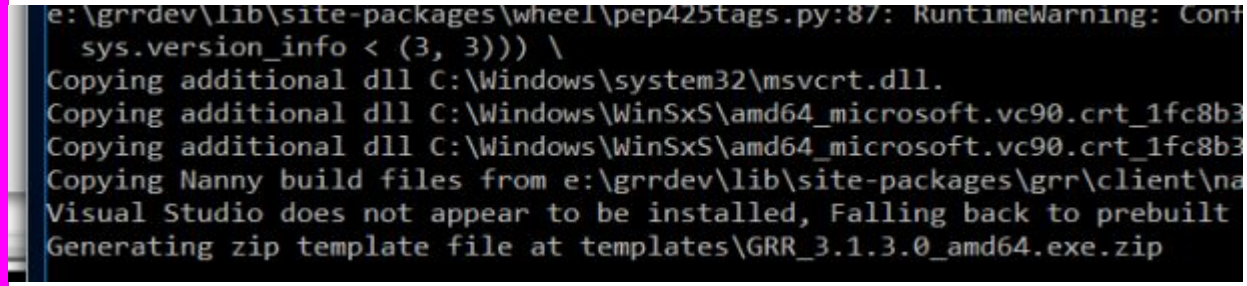
GRR Will normally download all the templates for all supported OSs and then repack them.

For this workshop we will do it by hand because this is a large download and we also want to learn how these are built.

Answer **no** to download the templates and **no** to repack them.

Building client templates from source.





We can just build the template and write it into the templates directory.

This template is for windows.

Try Out

Anatomy of a template.

```
(GRRDev) E:\>python -m zipfile -l templates\GRR_3.1.3.0_amd64.exe.zip
File Name                                             Modified             Size
./                                                2016-08-08 16:39:18            0
Include/                                          2016-08-08 16:39:14            0
pytz/                                             2016-08-08 16:39:16            0
build.yaml                                        2016-08-08 16:39:16          432
bz2.pyd                                           2016-08-08 16:11:22        94208
cryptography.hazmat.bindings._constant_time.pyd   2016-08-08 16:11:22         8192
cryptography.hazmat.bindings._openssl.pyd         2016-08-08 16:11:24      2809856
cryptography.hazmat.bindings._padding.pyd         2016-08-08 16:11:22         8704
grr-client.exe                                    2016-08-08 16:39:12      3949478
grr-client.exe.manifest                           2016-08-08 16:39:12         1011
GRRservice.exe                                    2016-08-08 16:39:18       129024
mfc90.dll                                         2016-05-05 00:19:40      5075968
```

A template is just a zip file with binaries in it.

build.yaml contains information about the template (architecture, OS, version etc).

You can not install the template yet - you need to **repack** it.

Repacking the template.

```
(GRRDev) E:\>grr_client_build --context "DebugClientBuild Context" repack --output_dir templates --template templates\GRR_3.1.3.0_amd64.exe.zip
INFO:2016-08-08 17:00:47,849 log:208] Starting GRR Prelogging buffer.
DEBUG:2016-08-08 17:00:48,052 __init__:70] lzma module is not available
DEBUG:2016-08-08 17:00:48,052 __init__:59] Registered VCS backend: git
DEBUG:2016-08-08 17:00:48,099 __init__:59] Registered VCS backend: hg
DEBUG:2016-08-08 17:00:48,099 pep425tags:80] Config variable 'Py_DEBUG' is unset, Python ABI tag may be incorrect
DEBUG:2016-08-08 17:00:48,099 pep425tags:80] Config variable 'WITH_PYMALLOC' is unset, Python ABI tag may be incorrect
DEBUG:2016-08-08 17:00:48,099 pep425tags:80] Config variable 'Py_UNICODE_SIZE' is unset, Python ABI tag may be incorrect
DEBUG:2016-08-08 17:00:48,099 pep425tags:80] Config variable 'Py_DEBUG' is unset, Python ABI tag may be incorrect
DEBUG:2016-08-08 17:00:48,099 pep425tags:80] Config variable 'WITH_PYMALLOC' is unset, Python ABI tag may be incorrect
DEBUG:2016-08-08 17:00:48,099 pep425tags:80] Config variable 'Py_UNICODE_SIZE' is unset, Python ABI tag may be incorrect
DEBUG:2016-08-08 17:00:48,148 __init__:59] Registered VCS backend: svn
DEBUG:2016-08-08 17:00:48,148 __init__:59] Registered VCS backend: bzr
INFO:2016-08-08 17:00:48,177 config_lib:1101] Loading configuration from e:\grrdev\install_data/etc/grr-server.yaml
WARNING:2016-08-08 17:00:48,177 config_lib:1042] Missing config definition for AdminUI.bind_port. This option is likely deprecated or renamed. Check the release notes.
WARNING:2016-08-08 17:00:48,177 config_lib:1042] Missing config definition for AdminUI.bind_port. This option is likely deprecated or renamed. Check the release notes.
DEBUG:2016-08-08 17:00:48,194 config_lib:660] Applying filter resource for install_data/etc.
INFO:2016-08-08 17:00:48,210 config_lib:1101] Loading configuration from e:\grrdev\install_data/etc/server.local.yaml
INFO:2016-08-08 17:00:48,210 config_lib:810] Configuration writeback is set to e:\grrdev\install_data/etc/server.local.yaml
DEBUG:2016-08-08 17:00:48,210 log:166] Initializing Logging subsystem.
DEBUG:2016-08-08 17:00:48,210 log:118] Will use logging engines ['file', 'stderr', 'event_log']
DEBUG:2016-08-08 17:00:48,210 config_lib:660] Applying filter env for WINDIR.
INFO:2016-08-08 17:00:48,210 log:148] Writing log file to C:\Windows\System32\logfiles\\GRR.log
DEBUG:2016-08-08 17:00:48,256 registry:147] Initializing CommunicatorInit
DEBUG:2016-08-08 17:00:48,272 registry:147] Initializing CommsInit
DEBUG:2016-08-08 17:00:48,272 registry:147] Initializing VFSInit
DEBUG:2016-08-08 17:00:48,272 registry:147] Initializing InitHook
Repacking template: templates\GRR_3.1.3.0_amd64.exe.zip
Using context: ['ClientBuilder Context', 'ClientBuilder Context', 'Platform:Windows', 'Arch:amd64', 'Arch:amd64', 'Platform:Windows', 'Target:Windows', 'Target:Windows', 'ClientBuilder Context', 'Platform:Windows', 'Arch:amd64', 'DebugClientBuild Context'] and labels: []
Loading configuration from c:\users\mic\appdata\local\temp\tmptkde_g\GRR.exe.yaml
Configuration writeback is set to c:\users\mic\appdata\local\temp\tmptkde_g\GRR.exe.yaml
Writing back configuration to file c:\users\mic\appdata\local\temp\tmptkde_g\GRR.exe.yaml
Deployable binary generated at templates\dbg_GRR_3.1.3.0_amd64.exe
Repacked into templates\dbg_GRR_3.1.3.0_amd64.exe
```

```
grr_client_build --context
"DebugClientBuild Context" repack
--output_dir templates --template
templates\GRR_3.1.3.0_amd64.exe.zip
```

Install the client.

```
(GRRDev) E:\>templates\dbg_GRR_3.1.3.0_amd64.exe
UnZipSFX 6.00 of 20 April 2009, by Info-ZIP (http://www.info-zip.org).
Payload Size 1049e4f from 28400
  inflating: C:\Users\mic\AppData\Local\Temp\TMPE010.tmp/GRR.exe
  inflating: C:\Users\mic\AppData\Local\Temp\TMPE010.tmp/GRR.exe.manifest
  inflating: C:\Users\mic\AppData\Local\Temp\TMPE010.tmp/GRRservice.exe
   creating: C:\Users\mic\AppData\Local\Temp\TMPE010.tmp/Include/
   creating: C:\Users\mic\AppData\Local\Temp\TMPE010.tmp/pytz/
  inflating: C:\Users\mic\AppData\Local\Temp\TMPE010.tmp/build.yaml
  inflating: C:\Users\mic\AppData\Local\Temp\TMPE010.tmp/bz2.pyd
```

Debug client will show exactly what is going on.

Non-debug client has silent install and runtime.

For now stop the service: `sc stop "grr monitor"`

```
C:\Windows\system32>sc stop "grr monitor"

SERVICE_NAME: grr monitor
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 3  STOP_PENDING
                                (STOPPABLE, NOT_PAUSA
WN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x3
        WAIT_HINT          : 0x0
```

Build client components


```
(GRRDev) E:\>grr_client_build build_components --output templates
Building component grr-chipsec-component, Version 1.2.2.2
Creating Virtual Env c:\users\mic\appdata\local\temp\tmpgewtv3
Built component:
message ClientComponent {
 build_system :   message Uname {
     architecture : u'64bit'
     machine : u'AMD64'
     pep425tag : u'cp27-cp27m-win_amd64'
     release : u'10'
     system : u'Windows'
     version : u'10.0.10240'
    }
 summary :   message ClientComponentSummary {
     cipher :   message SymmetricCipher {
         _algorithm : AES128CBC
         _iv : EncryptionKey:
            EncryptionKey (6fbbab40718eb31b917477f152970fb1a23872fba8b9a8601e4202
         _key : EncryptionKey:
            EncryptionKey (0fba2e0a7b5cc02c6a850f4ad2a0480c03f1387833d5e18139dcd4
        }
     modules : [
       u'grr_rekall'
       u'memory'
       u'rekall_types'
       u'rekall_pb2'
      ]
     name : u'grr-rekall'
     version : u'1.5.3rc2'
    }
}
Component saved as templates\grr-rekall_1.5.3rc2_cp27-cp27m-win_amd64.bin
```

Upload client component

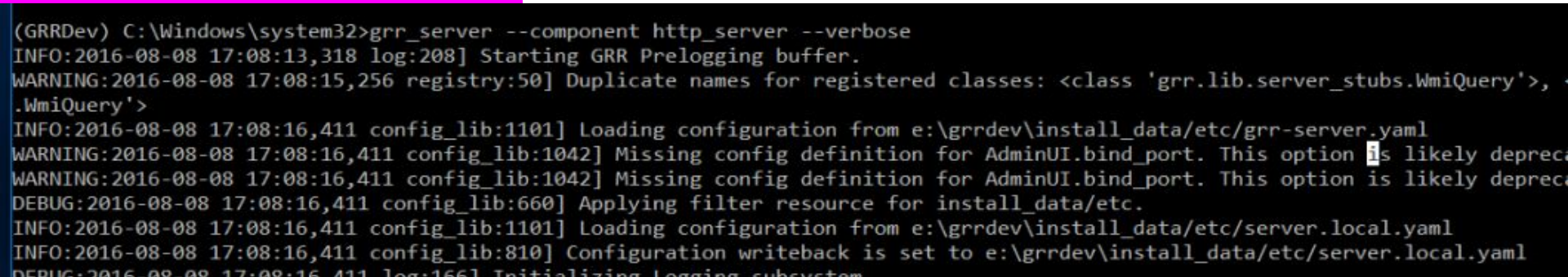
- Components are pushed to the clients.
- Components are architecture dependent.
  - File name indicates the architecture.
- Clients will refuse to run components which are not properly signed.
- grr_config_updater will sign and upload the component.

```
(GRRDev) E:\>grr_config_updater upload_component templates\grr-rekall_1.5.3rc2_cp27-cp27m-win_amd64.bin
Using configuration <GrrConfigManager  file="e:\grrdev\install_data/etc/grr-server.yaml"  file="e:\grrdev
Signing and uploading component templates\grr-rekall_1.5.3rc2_cp27-cp27m-win_amd64.bin
Opened component grr-rekall.
Storing component summary at aff4:/config/components/grr-rekall_1.5.3rc2
Storing signed component at aff4:/web/static/components/57a97b15f53c1aad/cp27-cp27m-win_amd64
```

Run the client, frontend and worker.

Use three different shells to run each component. Add the --verbose flag to see exactly what each component is doing.

Don't forget to activate the virtual env in each new terminal.



Start debug client by hand

The client runs with its own configuration file.

Windows clients are configured through the registry.

```
C:\Windows\system32>\Windows\System32\grr\3.1.3.0\GRR.exe --config \Windows\System32\grr\3.1.3.0\GRR.exe.yaml
INFO:2016-08-08 23:59:06,976 log:208] Starting GRR Prelogging buffer.
INFO:2016-08-08 23:59:07,101 config_lib:1101] Loading configuration from \Windows\System32\grr\3.1.3.0\GRR.exe.yaml
INFO:2016-08-08 23:59:07,101 config_lib:1101] Loading configuration from \Windows\System32\grr\3.1.3.0\build.yaml
INFO:2016-08-08 23:59:07,101 config_lib:1101] Loading configuration from reg://HKEY_LOCAL_MACHINE/Software\GRR
INFO:2016-08-08 23:59:07,101 config_lib:810] Configuration writeback is set to reg://HKEY_LOCAL_MACHINE/Software\GRR
DEBUG:2016-08-08 23:59:07,101 log:166] Initializing Logging subsystem.
DEBUG:2016-08-08 23:59:07,118 log:118] Will use logging engines ['file', 'stderr', 'event_log']
```

The end.