Forensic Analysis of Windows User Space Applications Through Heap Allocations

Michael Cohen
Google Inc.
Zurich, Switzerland

Third International Workshop on
Security and Forensics in Communication Systems
SFCS 2015

Why Userspace analysis?

- Forensically very valuable:
  - Users interact directly with applications.
  - Applications interact with the OS kernel.
  - Therefore we can sometimes infer user activity by OS kernel evidence but not always:
    - e.g. user chats on IRC
      - Sockets, Connections, network packets
      - Strings in IRC process - no context!

Challenges for user-space analysis

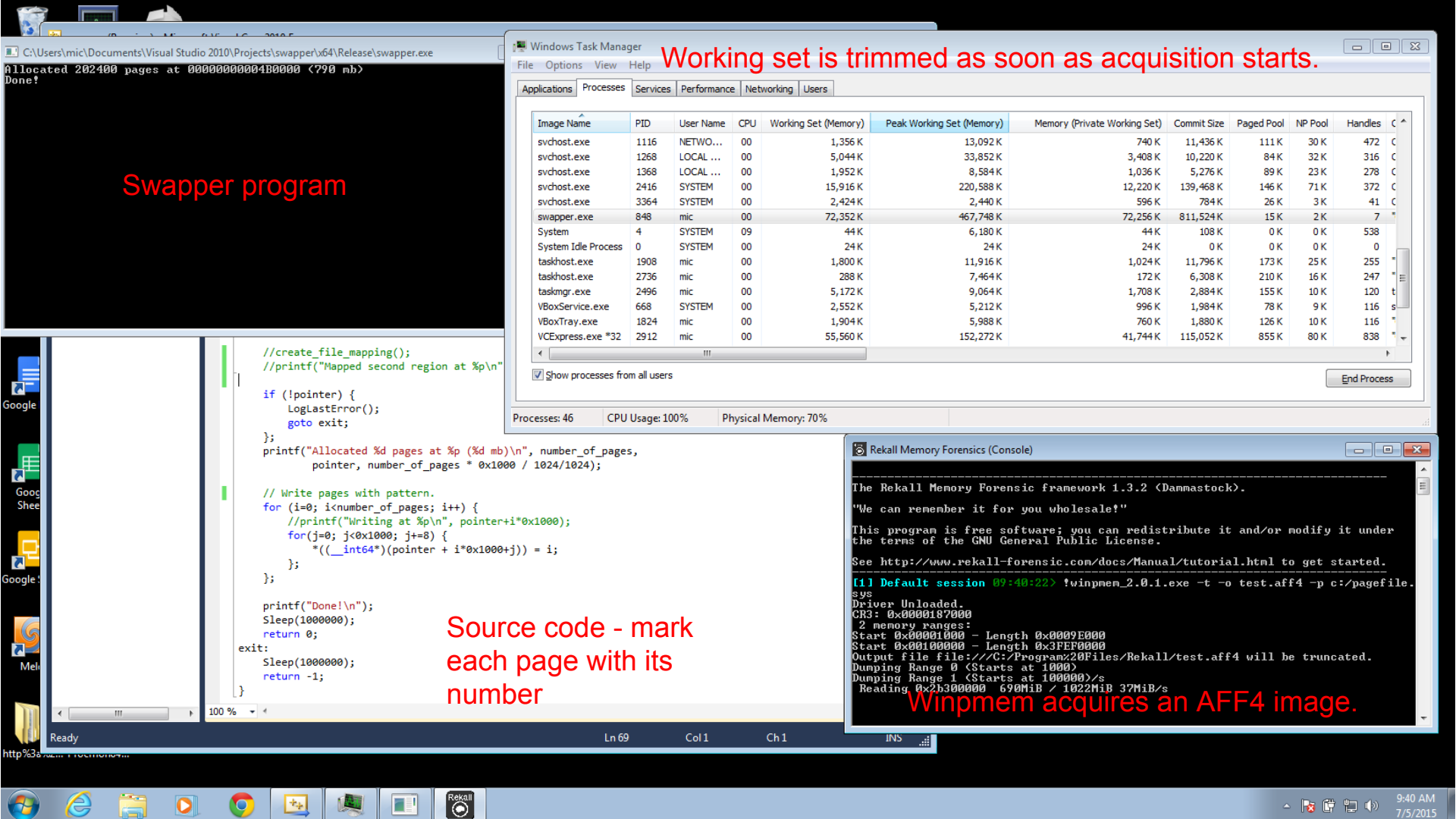
- So many user space applications - manual reversing just does not scale.
  - Lots of attention on reversing malware but there are many regular apps that are interesting too.
- User space memory is often paged and address translation is more complex.
  - Current tools and techniques are unable to resolve user space memory from Prototype PTEs or the Pagefile

Challenges for user-space analysis

- Why is page translation in userspace fairly complex?
  - Have to consider shared memory (Prototype PTEs).
  - Some memory forensic tools are extremely buggy:
    - Associate random data with the content of user space memory. (Very dangerous from an evidentiary perspective.).
  - Not a lot of tool testing or verification going on in Memory Forensics.

Experiment

- Small test program:
  - VirtualAlloc a large region (around 800Mb).
  - Mark each page with its sequence number - we can find the page in physical memory.
  - Sleep. Gives us plenty of time to acquire memory.
- What do we expect?
  - A VAD region for the allocated region.
  - When dumping the VAD region we expect marked pages in sequence (0, 1, 2, 3 etc).


C:\Users\mic\Documents\Visual Studio 2010\Projects\swapper\x64\Release\swapper.exe
Allocated 202400 pages at 00000000004B0000 (790 mb)
Done!
Swapper program
Windows Task Manager
File Options View Help
Working set is trimmed as soon as acquisition starts.
Applications Processes Services Performance Networking Users
Image Name PID User Name CPU Working Set (Memory) Peak Working Set (Memory) Memory (Private Working Set) Commit Size Paged Pool NP Pool Handles
svchost.exe 1116 NETWO... 00 1,356 K 13,092 K 740 K 11,436 K 111 K 30 K 472
svchost.exe 1268 LOCAL ... 00 5,044 K 33,852 K 3,408 K 10,220 K 84 K 32 K 316
svchost.exe 1368 LOCAL ... 00 1,952 K 8,584 K 1,036 K 5,276 K 89 K 23 K 278
svchost.exe 2416 SYSTEM 00 15,916 K 220,588 K 12,220 K 139,468 K 146 K 71 K 372
svchost.exe 3364 SYSTEM 00 2,424 K 2,440 K 596 K 784 K 26 K 3 K 41
swapper.exe 848 mic 00 72,352 K 467,748 K 72,256 K 811,524 K 15 K 2 K 7
System 4 SYSTEM 09 44 K 6,180 K 44 K 108 K 0 K 0 K 538
System Idle Process 0 SYSTEM 00 24 K 24 K 24 K 0 K 0 K 0 K 0
taskhost.exe 1908 mic 00 1,800 K 11,916 K 1,024 K 11,796 K 173 K 25 K 255
taskhost.exe 2736 mic 00 288 K 7,464 K 172 K 6,308 K 210 K 16 K 247
taskmgr.exe 2496 mic 00 5,172 K 9,064 K 1,708 K 2,884 K 155 K 10 K 120
VBoxService.exe 668 SYSTEM 00 2,552 K 5,212 K 996 K 1,984 K 78 K 9 K 116
VBoxTray.exe 1824 mic 00 1,904 K 5,988 K 760 K 1,880 K 126 K 10 K 116
VCExpress.exe *32 2912 mic 00 55,560 K 152,272 K 41,744 K 115,052 K 855 K 80 K 838
Show processes from all users
End Process
Processes: 46 CPU Usage: 100% Physical Memory: 70%
//create_file_mapping();
//printf("Mapped second region at %p\n"
if (!pointer) {
LogLastError();
goto exit;
};
printf("Allocated %d pages at %p (%d mb)\n", number_of_pages,
pointer, number_of_pages * 0x1000 / 1024/1024);
// Write pages with pattern.
for (i=0; i<number_of_pages; i++) {
//printf("Writing at %p\n", pointer+i*0x1000);
for(j=0; j<0x1000; j+=8) {
((__int64)(pointer + i*0x1000+j)) = i;
};
};
printf("Done!\n");
Sleep(1000000);
return 0;
exit:
Sleep(1000000);
return -1;
}
Source code - mark each page with its number
100 %
Ready Ln 69 Col 1 Ch 1 INS
Rekall Memory Forensics (Console)
The Rekall Memory Forensic framework 1.3.2 (Dammastock).
"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 09:40:22> !winpmem_2.0.1.exe -t -o test.aff4 -p c:/pagefile.sys
Driver Unloaded.
CR3: 0x0000187000
2 memory ranges:
Start 0x00001000 - Length 0x0009E000
Start 0x00100000 - Length 0x3FEF0000
Output file file:///C:/Program%20Files/Rekall/test.aff4 will be truncated.
Dumping Range 0 (Starts at 1000)
Dumping Range 1 (Starts at 100000)/s
Reading 0x2b300000 690MiB / 1022MiB 37MiB/s
Winpmem acquires an AFF4 image.
9:40 AM
7/5/2015

Let's inspect the VAD

```
[1] test.aff4 16:39:42> vad proc_regex="swapper"
----------------------> vad(proc_regex="swapper")
DEBUG:rekall.1:Running plugin (vad) with args (()) kwargs ({'proc_regex': 'swapper'})
************************************************
Pid: 4092 swapper.exe
     VAD       lev   Start Addr      End Addr     com   ------- ------       Protect        Filename
-------------- --- -------------- -------------- ------                  ------------------- --------
0xfa8001743ba0   5 0x000000010000 0x00000001ffff      0 Mapped          READWRITE
0xfa80014e2950   4 0x000000020000 0x00000002ffff      0 Mapped          READWRITE
0xfa8001f10380   5 0x000000030000 0x000000033fff      0 Mapped          READONLY
0xfa8001b80250   3 0x000000040000 0x000000040fff      0 Mapped          READONLY
0xfa80028ec170   5 0x000000050000 0x000000050fff      1 Private         READWRITE
0xfa80014e2770   4 0x000000060000 0x0000000c6fff      0 Mapped          READONLY            \Windows\System32\locale.nls
0xfa80016f7c70   5 0x0000000d0000 0x0000000fffff      0 Mapped          READONLY            \Windows\notepad.exe
0xfa80027afda0   2 0x000000130000 0x00000022ffff      6 Private         READWRITE
0xfa80014092f0   5 0x000000230000 0x0000002effff    192 Mapped          WRITECOPY           \Windows\System32\en-US\KernelBase.dll.mui
0xfa800303f590   4 0x000000390000 0x00000039ffff      9 Private         READWRITE
0xfa8002cb9ac0   3 0x0000004f0000 0x000031b8ffff 202400 Private         READWRITE

0xfa8000dcf2b0   4 0x0000738a0000 0x000073971fff     11 Mapped  Exe    EXECUTE_WRITECOPY   \Windows\System32\msvcr100.dll
0xfa80014e25b0   5 0x000077880000 0x00007799efff      4 Mapped  Exe    EXECUTE_WRITECOPY   \Windows\System32\kernel32.dll
0xfa8002c87280   1 0x000077aa0000 0x000077c47fff     12 Mapped  Exe    EXECUTE_WRITECOPY   \Windows\System32\ntdll.dll
0xfa80014d0680   4 0x00007efe0000 0x00007f0dffff      0 Mapped          READONLY
0xfa80016dc2b0   3 0x00007f0e0000 0x00007ffdffff      0 Private         READONLY
0xfa8002ce1890   2 0x00007ffe0000 0x00007fefffff     -1 Private         READONLY
0xfa8001797ef0   5 0x00013f670000 0x00013f676fff      3 Mapped  Exe    EXECUTE_WRITECOPY   \Users\mic\Documents\Visual Studio 2010\Projects\swapper\x64\Release\swapp
r.exe
0xfa80014c5280   4 0x07fefd880000 0x07fefd8ebfff      3 Mapped  Exe    EXECUTE_WRITECOPY   \Windows\System32\KernelBase.dll
0xfa8001ec0b80   5 0x07feffdc0000 0x07feffdc0fff      0 Mapped  Exe    EXECUTE_WRITECOPY   \Windows\System32\apisetschema.dll
0xfa8001bce2e0   3 0x07fffffb0000 0x07fffffd2fff      0 Mapped          READONLY
0xfa80014a51e0   4 0x07fffffdb000 0x07fffffdbfff      1 Private         READWRITE
0xfa8001df3220   5 0x07fffffde000 0x07fffffdffff      2 Private         READWRITE
```

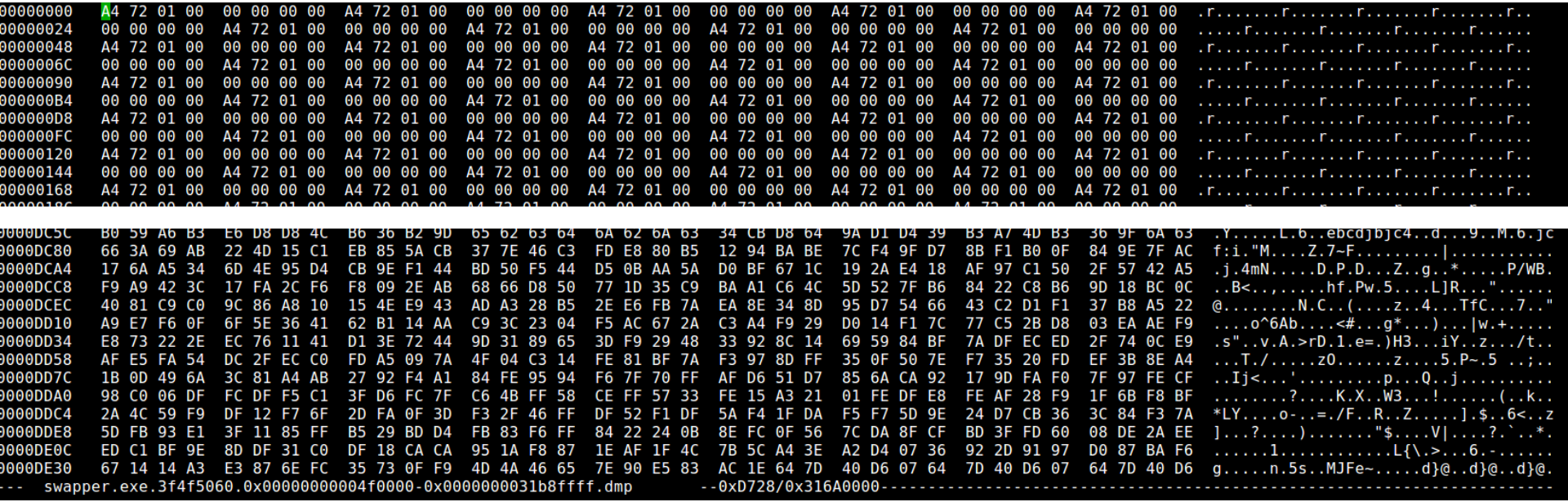
Let's dump out the allocated region

Export the image for Volatility (It can not process AFF4 images)

Latest commit Jun 23 2015: Run vaddump plugin to extract VAD regions.

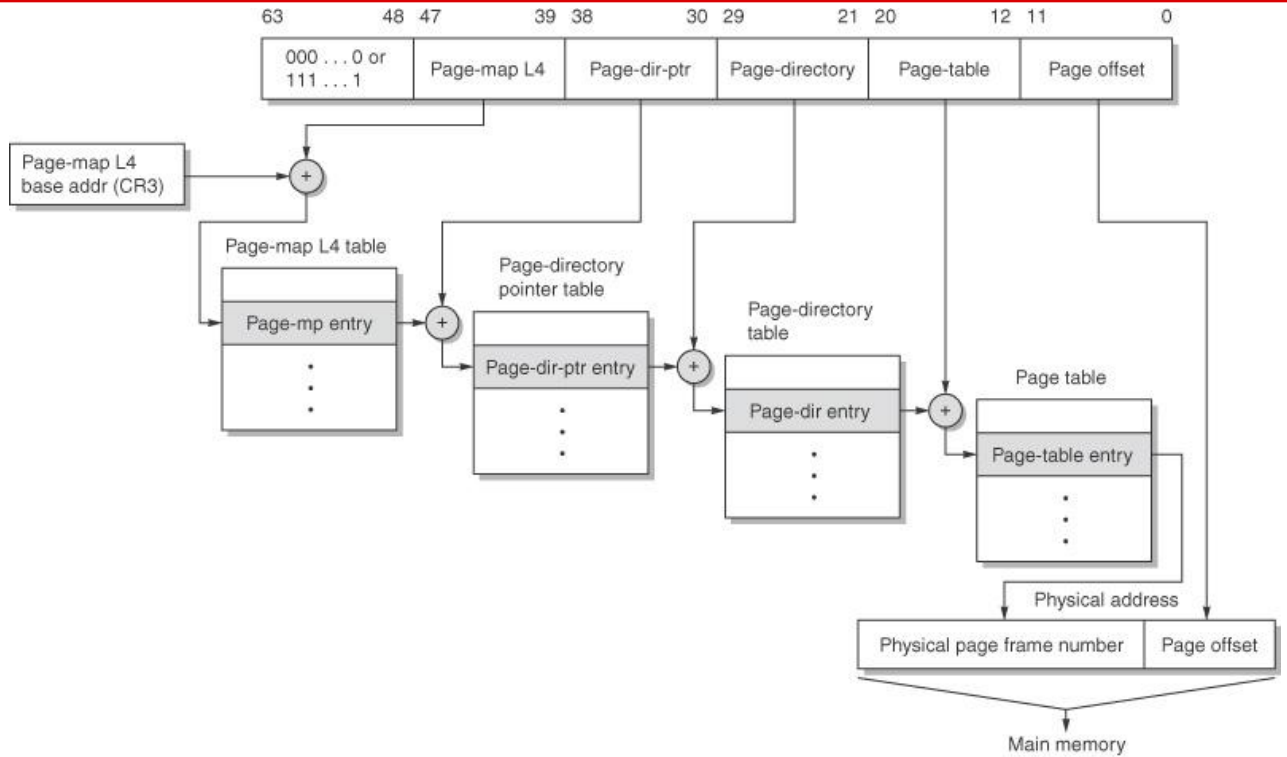
```
$ ./vol.py --profile Win7SP1x64 -f /tmp/image.raw vaddump -p 4092 -D /tmp/
Volatility Foundation Volatility Framework 2.4
*** Failed to import volatility.plugins.mimikatz (ImportError: No module named construct)
*** Failed to import volatility.plugins.dumpcerts (NameError: name 'yara' is not defined)
*** Failed to import volatility.plugins.linux.netscan (ImportError: No module named yara)
Pid        Process              Start              End                Result
---------- -------------------- ------------------ ------------------ ------
      4092 swapper.exe          0x0000000077aa0000 0x0000000077c47fff /tmp/swapper.exe.3f4f5060.0x0000000077aa0000-0x0000000077c47fff.dmp
      4092 swapper.exe          0x0000000000130000 0x000000000022ffff /tmp/swapper.exe.3f4f5060.0x0000000000130000-0x000000000022ffff.dmp
      4092 swapper.exe          0x0000000000040000 0x0000000000040fff /tmp/swapper.exe.3f4f5060.0x0000000000040000-0x0000000000040fff.dmp
      4092 swapper.exe          0x0000000000020000 0x0000000000002fff /tmp/swapper.exe.3f4f5060.0x0000000000020000-0x0000000000002fff.dmp
      4092 swapper.exe          0x0000000000010000 0x000000000001ffff /tmp/swapper.exe.3f4f5060.0x0000000000010000-0x000000000001ffff.dmp
      4092 swapper.exe          0x0000000000030000 0x0000000000033fff /tmp/swapper.exe.3f4f5060.0x0000000000030000-0x0000000000033fff.dmp
      4092 swapper.exe          0x0000000000060000 0x00000000000c6fff /tmp/swapper.exe.3f4f5060.0x0000000000060000-0x00000000000c6fff.dmp
      4092 swapper.exe          0x0000000000050000 0x0000000000050fff /tmp/swapper.exe.3f4f5060.0x0000000000050000-0x0000000000050fff.dmp
      4092 swapper.exe          0x00000000000d0000 0x00000000000fffff /tmp/swapper.exe.3f4f5060.0x00000000000d0000-0x00000000000fffff.dmp
      4092 swapper.exe          0x00000000004f0000 0x0000000031b8ffff /tmp/swapper.exe.3f4f5060.0x00000000004f0000-0x0000000031b8ffff.dmp
      4092 swapper.exe          0x0000000000390000 0x000000000039ffff /tmp/swapper.exe.3f4f5060.0x0000000000390000-0x000000000039ffff.dmp
      4092 swapper.exe          0x0000000000230000 0x000000000023ffff /tmp/swapper.exe.3f4f5060.0x0000000000230000-0x000000000023ffff.dmp
      4092 swapper.exe          0x00000000003f0000 0x00000000004effff /tmp/swapper.exe.3f4f5060.0x00000000003f0000-0x00000000004effff.dmp
      4092 swapper.exe          0x000000000738a000 0x0000000073971fff /tmp/swapper.exe.3f4f5060.0x000000000738a000-0x0000000073971fff.dmp
      4092 swapper.exe          0x0000000031b90000 0x000000000324ffff /tmp/swapper.exe.3f4f5060.0x0000000031b90000-0x000000000324ffff.dmp
      4092 swapper.exe          0x0000000077880000 0x000000007799efff /tmp/swapper.exe.3f4f5060.0x0000000077880000-0x000000007799efff.dmp
      4092 swapper.exe          0x000000007ffe0000 0x000000007ffeffff /tmp/swapper.exe.3f4f5060.0x000000007ffe0000-0x000000007ffeffff.dmp
      4092 swapper.exe          0x000000007f0e0000 0x000000007ffdffff /tmp/swapper.exe.3f4f5060.0x000000007f0e0000-0x000000007ffdffff.dmp
      4092 swapper.exe          0x000000007efe0000 0x000000007f0dffff /tmp/swapper.exe.3f4f5060.0x000000007efe0000-0x000000007f0dffff.dmp
      4092 swapper.exe          0x000007fffffb0000 0x000007fffffd2fff /tmp/swapper.exe.3f4f5060.0x000007fffffb0000-0x000007fffffd2fff.dmp
      4092 swapper.exe          0x000007fefd880000 0x000007fefd8ebfff /tmp/swapper.exe.3f4f5060.0x000007fefd880000-0x000007fefd8ebfff.dmp
      4092 swapper.exe          0x000000013f670000 0x000000013f676fff /tmp/swapper.exe.3f4f5060.0x000000013f670000-0x000000013f676fff.dmp
```

Lets take a look



Volatility: Many of the pages dumped are random garbage. Therefore Volatility can not handle user space memory reliably.

Address Translation Basics



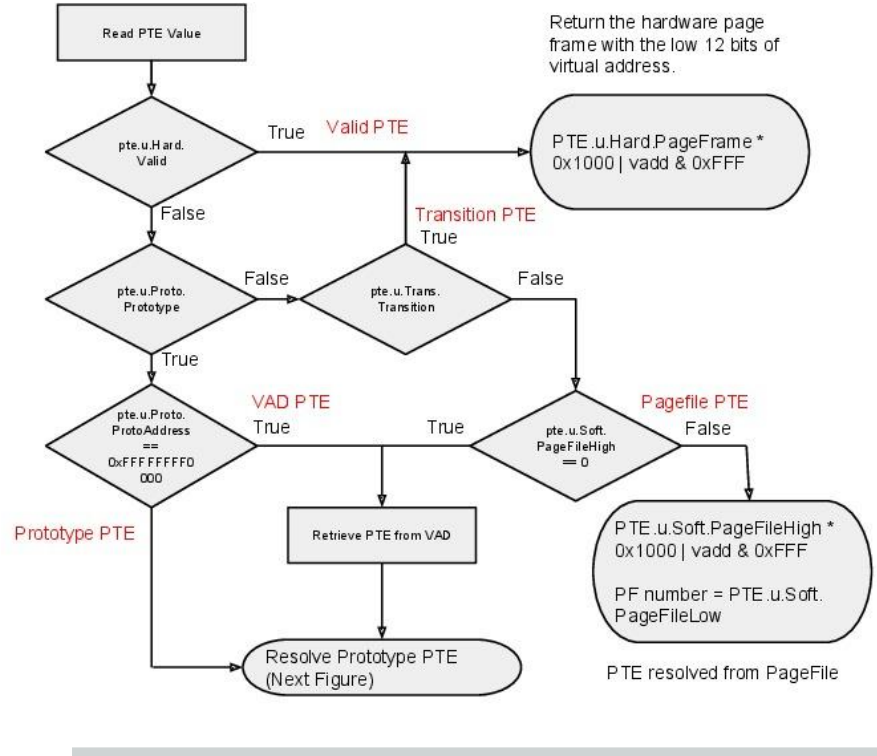
Google | Rekall

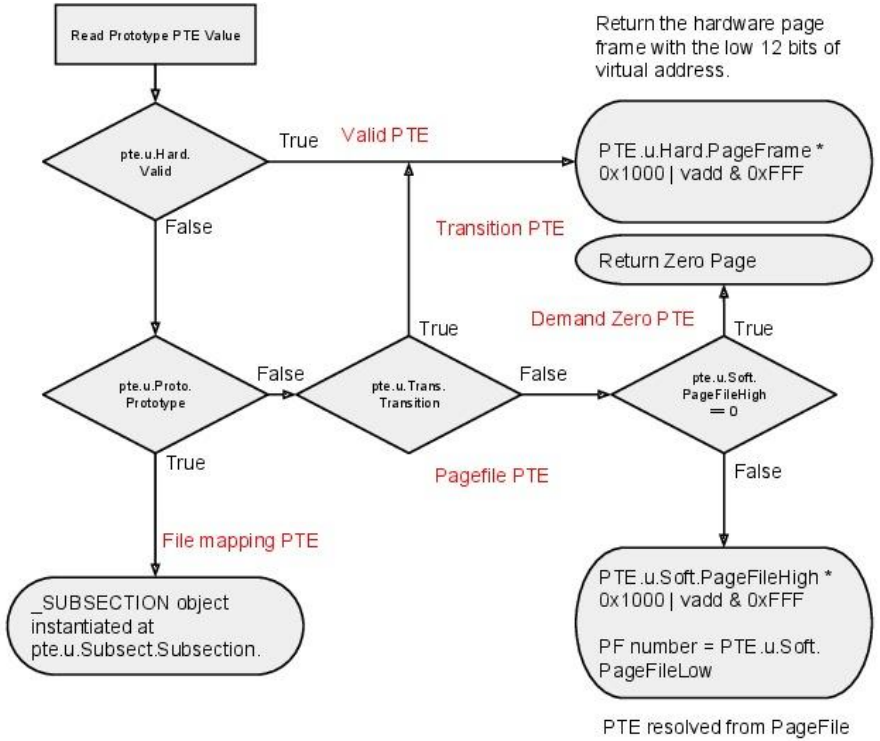
What if PTE bit 0 is unset?

Pagefault - Hardware calls into the OS kernel to resolve the mapping.

Kernel may use any of the other bits in the PTE as it wants. They are OS specific and have different meaning in different OS.

PTE resolution up on pagefault





Implementing OS specific translation

- We implemented the above algorithms in Rekall - the most advanced, open source, memory analysis framework.
- Also implemented plugins to inspect and verify state of PTE so we can double check the translation process.

Typical translation

```
[1] test.aff4 17:22:10> vtop 0x000004ff0000
----------------------> vtop(0x000004ff0000)
DEBUG:rekall.1:Running plugin (vtop) with args ((83820544,)) kwargs ({})

****************** 0x4ff0000 ******************
Virtual 0x000004ff0000 Page Directory 0x23d8c000
pml4e@ 0x23d8c000 = 0x1540000021497867
pdpte@ 0x21497000 = 0x11000003581b867
pde@ 0x3581b138 = 0xa30000024d7f847
pte@ 0x24d7ff80 = 0x22ce200000080

PTE Contains 0x22ce200000080
PTE Type: Pagefile
[_MMPTE_SOFTWARE Soft] @ 0x24D7FF80
  0x00 InStore                [BitField(22-23):InStore]: 0x00000000
  0x00 PageFileHigh           [BitField(32-64):PageFileHigh]: 0x00022CE2
  0x00 PageFileLow            [BitField(1-5):PageFileLow]: 0x00000000
  0x00 Protection             [Enumeration:Enumeration]: 0x00000004 (MM_READWRITE)
  0x00 Prototype              [BitField(10-11):Prototype]: 0x00000000
  0x00 Reserved               [BitField(23-32):Reserved]: 0x00000000
  0x00 Transition             [BitField(11-12):Transition]: 0x00000000
  0x00 UsedPageTableEntries   [BitField(12-22):UsedPageTableEntries]: 0x00000000
  0x00 Valid                  [BitField(0-1):Valid]: 0x00000000
Physical Address 0x859c4000 @ aff4://0603f7cb-b114-4e8c-b566-f43d47ab9fee/c:/pagefile.sys (Mapped 0x62ce2000)
```

Private mapping in pagefile

Resolve the address of the page in the pagefile (stored in the AFF4 Volume)

More complex example

```
[1] test.aff4 17:22:20> vtop 0x0000004f0000
----------------------> vtop(0x0000004f0000)
DEBUG:rekall.1:Running plugin (vtop) with args ((5177344,)) kwargs ({})

******************** 0x4f0000 ********************
Virtual 0x0000004f0000 Page Directory 0x23d8c000
pml4e@ 0x23d8c000 = 0x1540000021497867
pdpte@ 0x21497000 = 0x11000003581b867
pde@ 0x3581b010 = 0x1a2000003f266886
PDE Resolution:
PTE Contains 0x1a2000003f266886
PTE Type: Transition
[_MMPTE_TRANSITION Trans] @ 0x3581B010
  0x00 CacheDisable     [BitField(4-5):CacheDisable]: 0x00000000
  0x00 Owner            [BitField(2-3):Owner]: 0x00000001
  0x00 PageFrameNumber  [BitField(12-48):PageFrameNumber]: 0x0003F266
  0x00 Protection       [Enumeration:Enumeration]: 0x00000004 (MM_READWRITE)
  0x00 Prototype        [BitField(10-11):Prototype]: 0x00000000
  0x00 Transition       [BitField(11-12):Transition]: 0x00000001
  0x00 Unused           [BitField(48-64):Unused]: 0x00001A20
  0x00 Valid            [BitField(0-1):Valid]: 0x00000000
  0x00 Write            [BitField(1-2):Write]: 0x00000001
  0x00 WriteThrough     [BitField(3-4):WriteThrough]: 0x00000000
normalized pde@ 0x3581b010 = 0x1a2000003f266807
pte@ 0x3f266780 = 0x3232800000080

PTE Contains 0x3232800000080
PTE Type: Pagefile
[_MMPTE_SOFTWARE Soft] @ 0x3F266780
  0x00 InStore              [BitField(22-23):InStore]: 0x00000000
  0x00 PageFileHigh         [BitField(32-64):PageFileHigh]: 0x00032328
  0x00 PageFileLow          [BitField(1-5):PageFileLow]: 0x00000000
  0x00 Protection           [Enumeration:Enumeration]: 0x00000004 (MM_READWRITE)
  0x00 Prototype            [BitField(10-11):Prototype]: 0x00000000
  0x00 Reserved             [BitField(23-32):Reserved]: 0x00000000
  0x00 Transition           [BitField(11-12):Transition]: 0x00000000
  0x00 UsedPageTableEntries [BitField(12-22):UsedPageTableEntries]: 0x00000000
  0x00 Valid                [BitField(0-1):Valid]: 0x00000000
Physical Address 0xa4650000 @ aff4://0603f7cb-b114-4e8c-b566-f43d47ab9fee/c:/pagefile.sys (Mapped 0x72328000)
```

PDE is in Transition. First resolve PDE to find PTE.

Shared memory (e.g. DLL)

```
[1] test.aff4 17:28:13> vtop 0x0000738a0000
----------------------> vtop(0x0000738a0000)
DEBUG:rekall.1:Running plugin (vtop) with args ((1938423808,)) kwargs ({})

****************** 0x738a0000 ******************
Virtual 0x0000738a0000 Page Directory 0x23d8c000
pml4e@ 0x23d8c000 = 0x1540000021497867
pdpte@ 0x21497008 = 0x15000000789f867
pde@ 0x789fce0 = 0x6e00000039d74847
pte@ 0x39d74500 = 0xf8a00a2f49280400

PTE Contains 0xf8a00a2f49280400
PTE Type: Prototype
[_MMPTE_PROTOTYPE Proto] @ 0x39D74500
  0x00 Protection    [Enumeration:Enumeration]: 0x00000000 (MM_ZERO_ACCESS)
  0x00 ProtoAddress  [BitField(16-64):ProtoAddress]: 0xF8A00A2F4928
  0x00 Prototype     [BitField(10-11):Prototype]: 0x00000001
  0x00 ReadOnly      [BitField(8-9):ReadOnly]: 0x00000000
  0x00 Unused0       [BitField(1-8):Unused0]: 0x00000000
  0x00 Unused1       [BitField(9-10):Unused1]: 0x00000000
  0x00 Valid         [BitField(0-1):Valid]: 0x00000000
Prototype PTE backed by file.
[_MMPTE_SUBSECTION Subsect] @ 0xF8A00A2F4928
  0x00 Protection         [Enumeration:Enumeration]: 0x00000001 (MM_READONLY)
  0x00 Prototype          [BitField(10-11):Prototype]: 0x00000001
  0x00 SubsectionAddress  [BitField(16-64):SubsectionAddress]: 0xFA80017347F0
  0x00 Unused0            [BitField(1-5):Unused0]: 0x00000000
  0x00 Unused1            [BitField(11-16):Unused1]: 0x00000000
  0x00 Valid              [BitField(0-1):Valid]: 0x00000000
Filename: \Windows\System32\msvcr100.dll
File Offset: 0 (           0x0)
Physical Address Invalid
```

Hardware PTE points to Prototype PTE

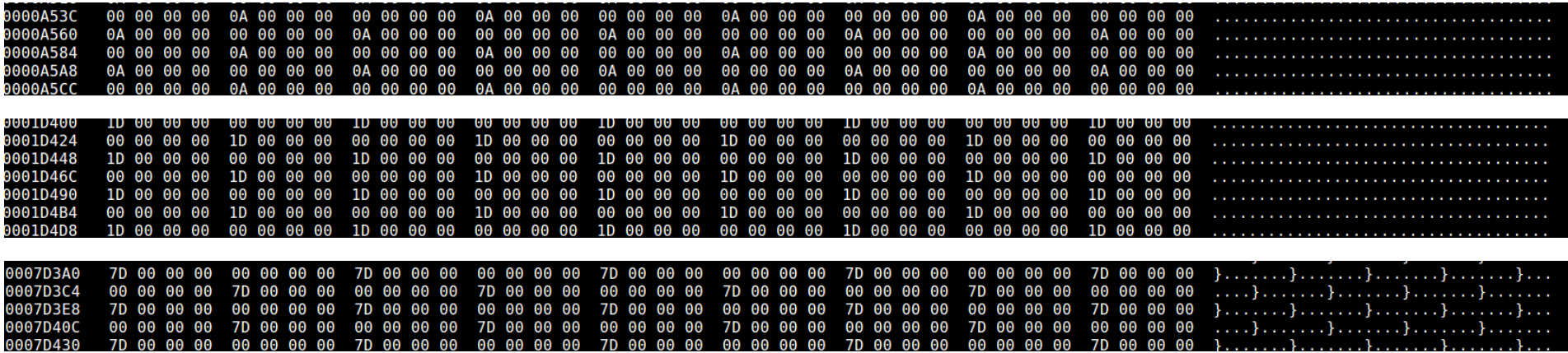
Prototype PTE points to file mapping (Subsection)

Examine our experiment

```
[1] test.aff4 17:31:50> vadmap proc_regex="swapper", start=0x0000004f000
---------------------> vadmap(proc_regex="swapper", start=0x0000004f000
**************************************************
Pid: 4092 swapper.exe
  Virt Addr        Length             Type          Comments
-------------- -------------- ------------------- --------
0x0000004f0000       0xc03d000 Pagefile            PF 0 @ 0x32328000
0x00000c52d000       0xbcd3000 Transition          PhysAS @ 0x28ea4000
0x000018200000        0x400000 Pagefile            PF 0 @ 0x2a058000
0x000018600000        0x200000 Transition          PhysAS @ 0x3ca8c000
0x000018800000        0x800000 Pagefile            PF 0 @ 0x4b9d5000
0x000019000000        0x200000 Transition          PhysAS @ 0xbaee000
0x000019200000        0x200000 Pagefile            PF 0 @ 0x312b7000
0x000019400000        0x200000 Transition          PhysAS @ 0x23f62000
0x000019600000        0x400000 Pagefile            PF 0 @ 0x160c000
0x000019a00000        0x200000 Transition          PhysAS @ 0x3fb28000
0x000019c00000        0x400000 Pagefile            PF 0 @ 0x42c32000
0x00001a000000        0x200000 Transition          PhysAS @ 0x285e7000
0x00001a200000       0x1c00000 Pagefile            PF 0 @ 0x19e1000
0x00001be00000        0x200000 Transition          PhysAS @ 0x2286000
0x00001c000000       0x2652000 Pagefile            PF 0 @ 0x41c49000
0x00001e652000        0x45c000 Transition          PhysAS @ 0x7814000
0x00001eaae000       0x1284000 Pagefile            PF 0 @ 0x1d1f3000
0x00001fd32000       0x1ea4000 Transition          PhysAS @ 0x210e6000
0x000021bd6000        0x1e2000 Pagefile            PF 0 @ 0x1d203000
0x000021db8000        0xf48000 Transition          PhysAS @ 0xe50d000
0x000022d00000        0xe1d000 Pagefile            PF 0 @ 0x356e9000
0x000023b1d000        0xf87000 Transition          PhysAS @ 0xa99d000
0x000024aa4000         0xc6000 Pagefile            PF 0 @ 0x488fb000
0x000024b6a000        0x480000 Transition          PhysAS @ 0x3bf1e000
0x000024fea000        0x215000 Pagefile            PF 0 @ 0x44b00000
0x0000251ff000        0xd16000 Transition          PhysAS @ 0xd277000
0x000025f15000        0x45f000 Pagefile            PF 0 @ 0x421e5000
0x000026374000        0xcb7000 Transition          PhysAS @ 0x32463000
0x00002702b000         0x7b000 Pagefile            PF 0 @ 0x44e6000
0x0000270a6000        0x167000 Transition          PhysAS @ 0x28b88000
0x00002720d000         0x3b000 Pagefile            PF 0 @ 0x20375000
0x000027248000       0x1e9b000 Transition          PhysAS @ 0x397f3000
0x0000290e3000         0x3a000 Pagefile            PF 0 @ 0x16f64000
0x00002911d000       0x581c000 Transition          PhysAS @ 0x117d6000
0x00002e939000         0x32000 Valid               PhysAS @ 0xdf58000
0x00002e96b000        0xb4f000 Transition          PhysAS @ 0xc23f000
0x00002f4ba000         0xdb000 Valid               PhysAS @ 0x25794000
0x00002f595000       0x19ba000 Transition          PhysAS @ 0x1ac2b000
0x000030f4f000        0xc41000 Valid               PhysAS @ 0x3c017000
0x000031b90000        0x960000 Demand Zero
0x0000738a0000         0x21000 File Mapping        \Windows\System32\msvcr100.dll (P)
0x0000738c1000          0x3000 Valid               PhysAS @ 0x176ed000
0x0000738c4000          0x1000 File Mapping        \Windows\System32\msvcr100.dll @ 0x23400 (P)
0x0000738c5000          0x1000 Valid               PhysAS @ 0x34d2e000
0x0000738c6000          0x6000 File Mapping        \Windows\System32\msvcr100.dll @ 0x25400 (P)
0x0000738cc000          0x3000 Valid               PhysAS @ 0x1ac37000
0x0000738cf000          0x1000 File Mapping        \Windows\System32\msvcr100.dll @ 0x2e400 (P)
0x0000738d0000          0x1000 Valid               PhysAS @ 0xe9e6000
0x0000738d1000          0xb000 File Mapping        \Windows\System32\msvcr100.dll @ 0x30400 (P)
0x0000738dc000          0x2000 Valid               PhysAS @ 0x3d72000
0x0000738de000         0x15000 File Mapping        \Windows\System32\msvcr100.dll @ 0x3d400 (P)
0x0000738f3000          0x1000 Valid               PhysAS @ 0x33eff000
0x0000738f4000         0x18000 File Mapping        \Windows\System32\msvcr100.dll @ 0x53400 (P)
0x00007390c000          0x2000 Valid               PhysAS @ 0x25bf9000
0x00007390e000          0x2000 File Mapping        \Windows\System32\msvcr100.dll @ 0x6d400 (P)
0x000073910000          0x1000 Valid               PhysAS @ 0x241000
0x000073911000          0x3000 File Mapping        \Windows\System32\msvcr100.dll @ 0x70400 (P)
0x000073914000          0x1000 Pagefile            PF 0 @ 0x323f9000
0x000073915000         0x1c000 File Mapping        \Windows\System32\msvcr100.dll @ 0x74400 (P)
0x000073931000          0x4000 Valid               PhysAS @ 0x365f000
0x000073935000         0x1d000 File Mapping        \Windows\System32\msvcr100.dll @ 0x93c00 (P)
0x000073952000          0x2000 Valid               PhysAS @ 0x20a37000
0x000073954000          0x3000 File Mapping        \Windows\System32\msvcr100.dll @ 0xb2800 (P)
0x000073957000          0x3000 Valid               PhysAS @ 0x2b7d5000
0x00007395a000         0x18000 File Mapping        \Windows\System32\msvcr100.dll @ 0xb5a00 (P)
0x000077880000         0x21000 Valid               PhysAS @ 0x1536000 (P)
0x0000778a1000          0x2000 Valid               PhysAS @ 0x222ed000
0x0000778a3000          0x1000 Valid               PhysAS @ 0x3fb6d000 (P)
0x0000778a4000          0x4000 Transition          PhysAS @ 0xde6e000 (P)
```

Page resolved through prototype
although it is actually still resident.

Use Rekall to dump VAD

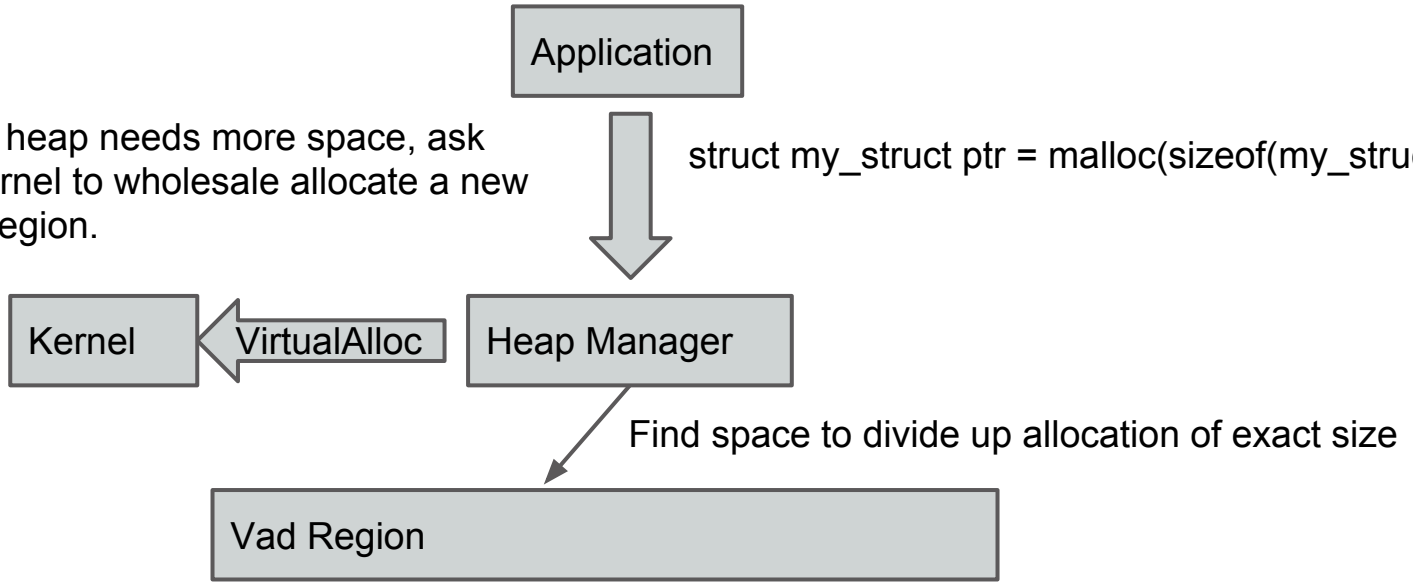


Vad region dumped accurately. All pages are correctly resolved. We have "perfect Rekall"!

What next?

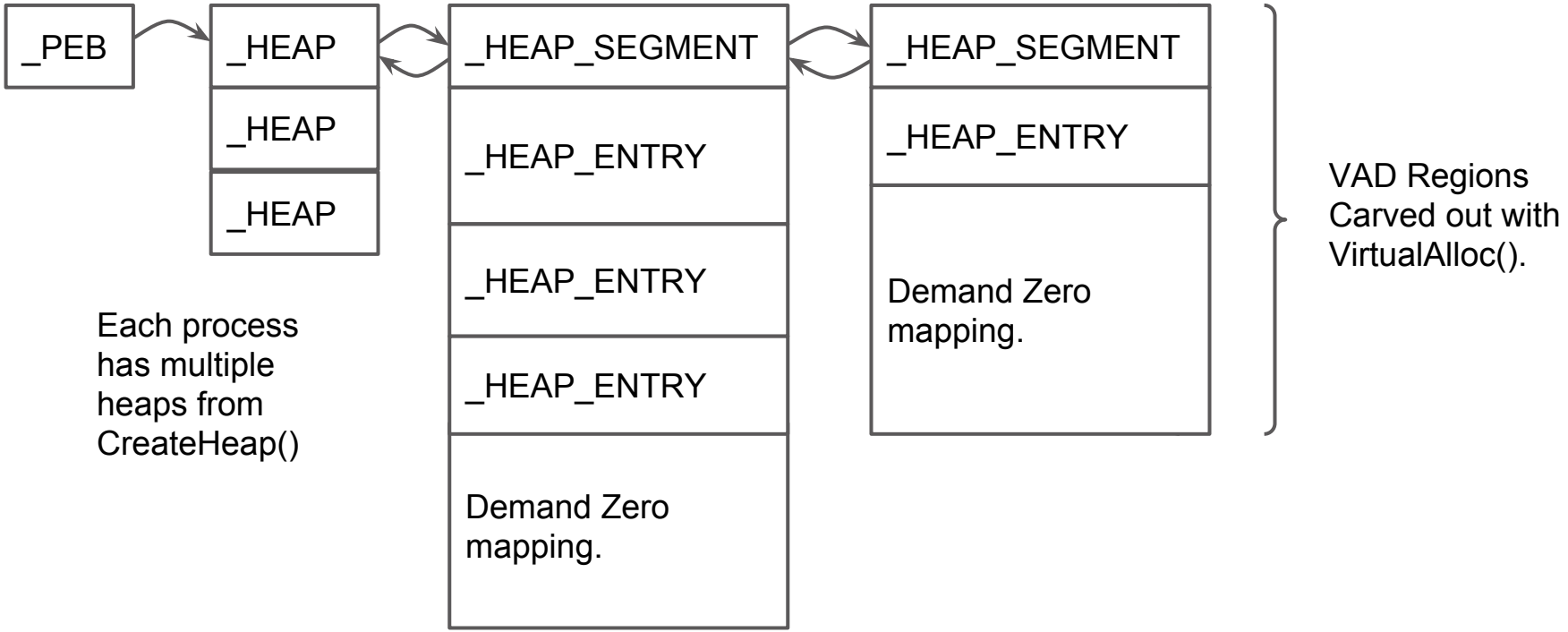
- So now we can reproduce userspace memory accurately what can we do?
  - Applications allocate memory using a heap allocator typically implemented by a library (e.g. MSVCRT).
  - Applications use higher level abstractions
    - Struct - represent similar objects (Size + Use)
    - Data structures:
      - Linked lists
      - Hash table
      - Strings (UTF 16 encoded)

Heap allocation in practice

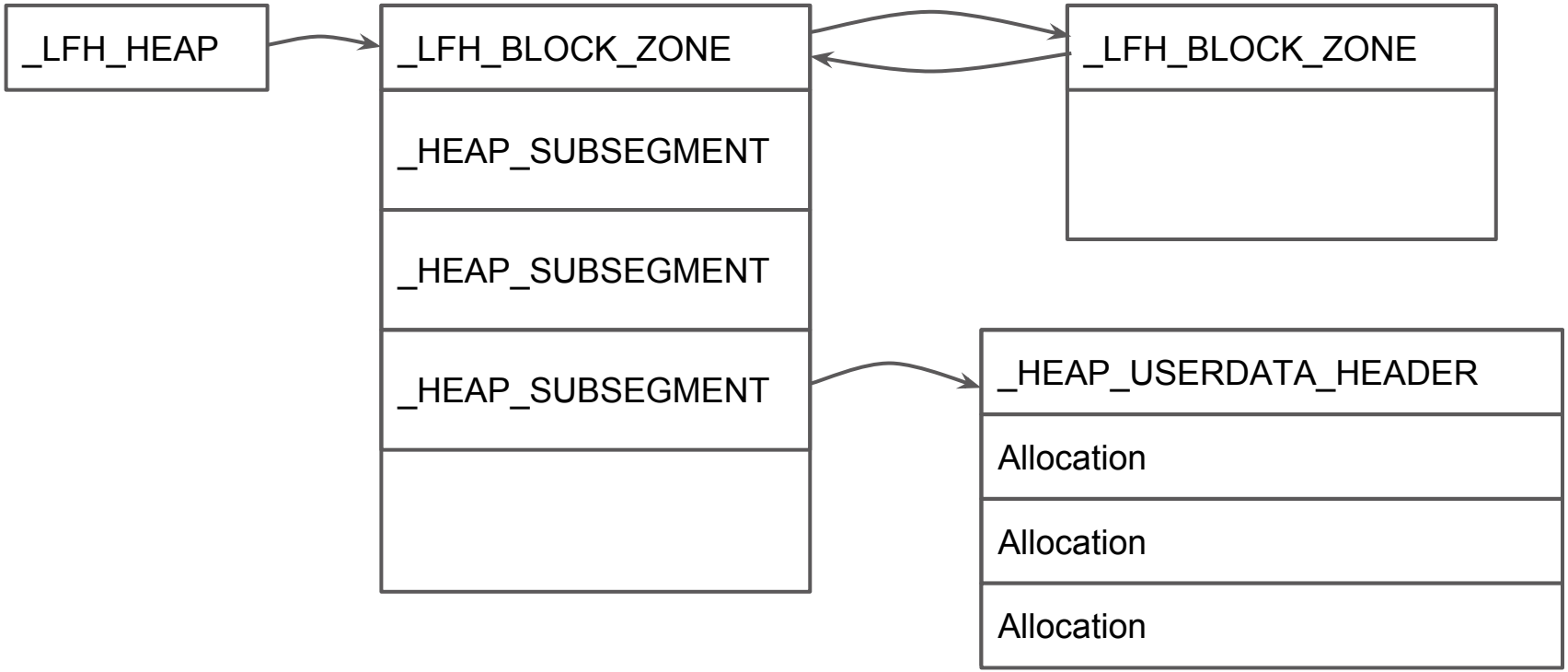


Point of view of Kernel - User allocations are large VAD regions.
Point of view of Application - User allocations are small precise allocations with their own implied purpose.

Backend Heap



Low Fragmentation Heap

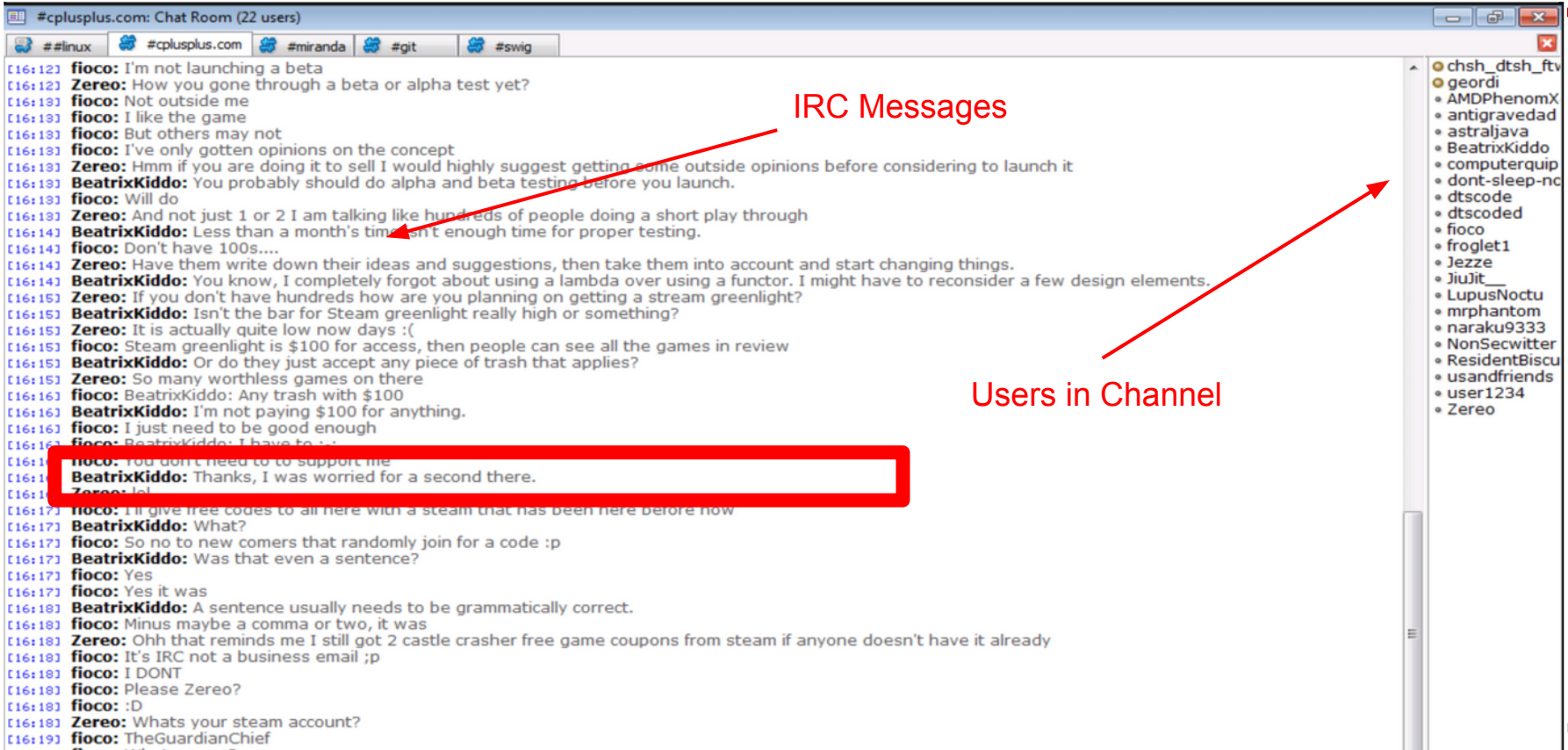


Google | Rekall

Reversing through heap analysis

- If we can split user space memory into precise allocations we can more easily see relationships between internal data structures.
- Sometime this avoids the need to reverse any code.

Example: Miranda IRC client



Pick a message and search for it

```
[1] MirandaTest.E01 17:53:43> grep "Thanks, I".encode("utf-16-le")
----------------------------> grep("Thanks, I".encode("utf-16-le"))
DEBUG:rekall.1:Running plugin (grep) with args (('T\x00h\x00a\x00n\x00k\x00s\x00,\x00 \x
DEBUG:rekall.1:Opened local file /usr/local/google/home/scudette/.rekall_cache/sessions/
    Offset                                      Data
-------------- ------------------------------------------------------------------- -------
     0x32ddd0 78 00 4b 00 69 00 64 00 64 00 6f 00 3a 00 20 00  x.K.i.d.d.o.:...
     0x32dde0 54 00 68 00 61 00 6e 00 6b 00 73 00 2c 00 20 00  T.h.a.n.k.s.,...
     0x32ddf0 49 00 20 00 77 00 61 00 73 00 20 00 77 00 6f 00  I...w.a.s...w.o.
    Offset                                      Data
-------------- ------------------------------------------------------------------- -------
     0x32fcf8 78 00 4b 00 69 00 64 00 64 00 6f 00 3a 00 20 00  x.K.i.d.d.o.:...
     0x32fd08 54 00 68 00 61 00 6e 00 6b 00 73 00 2c 00 20 00  T.h.a.n.k.s.,...
     0x32fd18 49 00 20 00 77 00 61 00 73 00 20 00 77 00 6f 00  I...w.a.s...w.o.
    Offset                                      Data
-------------- ------------------------------------------------------------------- -------
     0x33e690 78 00 4b 00 69 00 64 00 64 00 6f 00 3a 00 20 00  x.K.i.d.d.o.:...
     0x33e6a0 54 00 68 00 61 00 6e 00 6b 00 73 00 2c 00 20 00  T.h.a.n.k.s.,...
     0x33e6b0 49 00 20 00 77 00 61 00 73 00 20 00 77 00 6f 00  I...w.a.s...w.o.
    Offset                                      Data
-------------- ------------------------------------------------------------------- -------
    0x42ad108 cf 94 b1 45 25 00 00 90 54 00 00 00 ba ba ba ab  ...E%...T.......
    0x42ad118 54 00 68 00 61 00 6e 00 6b 00 73 00 2c 00 20 00  T.h.a.n.k.s.,...
    0x42ad128 49 00 20 00 77 00 61 00 73 00 20 00 77 00 6f 00  I...w.a.s...w.o.
```

View the string allocation

- Rekall shows the string is allocated from front-end allocator, with size 112 bytes.
  - Appears to have 8 bytes preamble (size + const).

```
[1] MirandaTest.aff4 21:32:05> show_allocation 0x42ad118
----------------------------> show_allocation(0x42ad118)
DEBUG:rekall.1:Running plugin (show_allocation) with args ((69914904,)) kwargs ({})
Address      0x42ad118 is 8 bytes into F allocation of size 112 (     0x42ad110 -      0x42ad180)
   Offset  Appears to be extra allocation data.  Data                                    Comment
---------- ----------------------------------------------- ----------------------------
0x42ad110 54 00 00 00 ba ba ba ab 54 00 68 00 61 00 6e 00  T.......T.h.a.n.
0x42ad120 6b 00 73 00 2c 00 20 00 49 00 20 00 77 00 61 00  k.s.,...I...w.a.
0x42ad130 73 00 20 00 77 00 6f 00 72 00 72 00 69 00 65 00  s...w.o.r.r.i.e.
0x42ad140 64 00 20 00 66 00 6f 00 72 00 20 00 61 00 20 00  d...f.o.r...a...
0x42ad150 73 00 65 00 63 00 6f 00 6e 00 64 00 20 00 74 00  s.e.c.o.n.d...t.
0x42ad160 68 00 65 00 72 00 65 00 2e 00 00 00 ba ba ba ab  h.e.r.e.........
0x42ad170 63 00 00 00 00 00 00 00 c8 94 b1 45 25 00 00 94  c..........E%...
                  Out<2> Plugin: show_allocation
```

Who refers to this string?

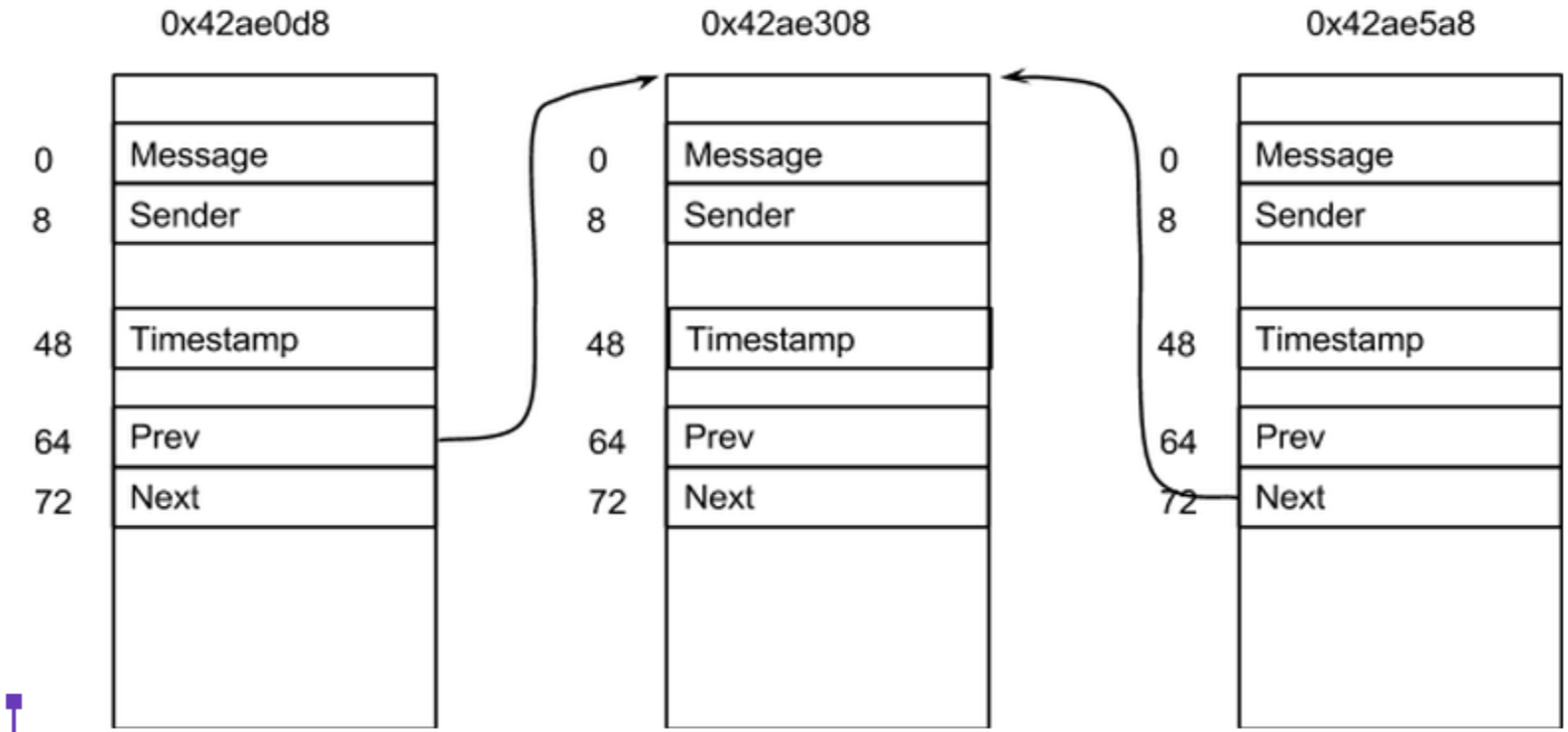
- References to start of string.
  - No references to start of alloc - only to string.

```
[1] MirandaTest.E01 18:32:05> show_referrer_alloc 0x42ad110
----------------------------> show_referrer_alloc(0x42ad110)
DEBUG:rekall.1:Running plugin (show_referrer_alloc) with args ((69914896,)) kwargs ({})
                   Out<11> Plugin: show_referrer_alloc
[1] MirandaTest.E01 18:32:08> show_referrer_alloc 0x42ad118
----------------------------> show_referrer_alloc(0x42ad118)
DEBUG:rekall.1:Running plugin (show_referrer_alloc) with args ((69914904,)) kwargs ({})
Address      0x42ae308 is 8 bytes into F allocation of size 112 (     0x42ae300 -      0x42ae370)
    Offset                                  Data                                                  Comment
------------ ------------------------------------------------------------------- -------------------------
     0x42ae300 50 00 00 00 ba ba ba ab 18 d1 2a 04 00 00 00 00  P.......  ..*.....  0x42ad118(112@0x42ad110)
     0x42ae310 08 43 27 04 00 00 00 00 00 00 00 00 00 00 00 00  .C'.....  ........  0x4274308(48@0x4274300)
     0x42ae320 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ........  ........
     0x42ae330 00 00 00 00 00 00 00 00 5e 80 90 55 00 00 00 00  ........  ^..U....
     0x42ae340 40 00 00 00 00 00 00 00 d8 e0 2a 04 00 00 00 00  @.......  ..*.....  0x42ae0d8(112@0x42ae0d0)
     0x42ae350 a8 e5 2a 04 00 00 00 00 ba ba ba ab 20 00 41 00  ..*.....  ......A.  0x42ae5a8(112@0x42ae5a0)
     0x42ae360 4e 00 53 00 49 00 00 00 e9 97 b1 45 25 00 00 94  N.S.I...  ...E%...
                   Out<12> Plugin: show_allocation
```

What does this struct refer to?

```
[1] MirandaTest.E01 18:51:14> show_allocation 0x42ad118, 0x4274308, 0x42ae0d8, 0x42ae5a8
----------------------------> show_allocation(0x42ad118, 0x4274308, 0x42ae0d8, 0x42ae5a8)
DEBUG:rekall.1:Running plugin (show_allocation) with args ((69914904, 69681928, 69918936, 69920168)) kwargs ({})
Address      0x42ad118 is 8 bytes into F allocation of size 112 (     0x42ad110 -      0x42ad180)
    Offset                                  Data                                        Comment
-------------- ------------------------------------------------------------------- ----------------------------------
     0x42ad110 54 00 00 00 ba ba ba ab 54 00 68 00 61 00 6e 00  T.......T.h.a.n.
     0x42ad120 6b 00 73 00 2c 00 20 00 49 00 20 00 77 00 61 00  k.s.,...I...w.a.
     0x42ad130 73 00 20 00 77 00 6f 00 72 00 72 00 69 00 65 00  s...w.o.r.r.i.e.
     0x42ad140 64 00 20 00 66 00 6f 00 72 00 20 00 61 00 20 00  d...f.o.r...a...
     0x42ad150 73 00 65 00 63 00 6f 00 6e 00 64 00 20 00 74 00  s.e.c.o.n.d...t.
     0x42ad160 68 00 65 00 72 00 65 00 2e 00 00 00 ba ba ba ab  h.e.r.e.........
     0x42ad170 63 00 00 00 00 00 00 00 c8 94 b1 45 25 00 00 94  c..........E%...
Address      0x4274308 is 8 bytes into F allocation of size 48 (     0x4274300 -      0x4274330)
    Offset                                  Data                                        Comment
-------------- ------------------------------------------------------------------- ----------------------------------
     0x4274300 1a 00 00 00 ba ba ba ab 42 00 65 00 61 00 74 00  ........B.e.a.t.
     0x4274310 72 00 69 00 78 00 4b 00 69 00 64 00 64 00 6f 00  r.i.x.K.i.d.d.o.
     0x4274320 00 00 ba ba ba ab 00 00 8f 4d b1 45 25 00 00 8a  .........M.E%...
Address      0x42ae0d8 is 8 bytes into F allocation of size 112 (     0x42ae0d0 -      0x42ae140)
    Offset                                  Data                                        Comment
-------------- ------------------------------------------------------------------- ----------------------------------
     0x42ae0d0 50 00 00 00 ba ba ba ab d8 fb 26 04 00 00 00 00  P.........&.....  0x426fbd8(96@0x426fbd0)
     0x42ae0e0 88 fc cf 01 00 00 00 00 00 00 00 00 00 00 00 00  ................  0x1cffc88(32@0x1cffc80)
     0x42ae0f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
     0x42ae100 00 00 00 00 00 00 00 00 50 80 90 55 00 00 00 00  ........P..U....
     0x42ae110 40 00 00 00 00 00 00 00 98 34 27 04 00 00 00 00  @........4'.....  0x4273498(112@0x4273490)
     0x42ae120 08 e3 2a 04 00 00 00 00 ba ba ba ab 00 00 00 00  ..*.............  0x42ae308(112@0x42ae300)
     0x42ae130 00 00 00 00 00 00 00 00 cc 97 b1 45 25 00 00 94  ...........E%...
Address      0x42ae5a8 is 8 bytes into F allocation of size 112 (     0x42ae5a0 -      0x42ae610)
    Offset                                  Data                                        Comment
-------------- ------------------------------------------------------------------- ----------------------------------
     0x42ae5a0 50 00 00 00 ba ba ba ab 98 62 2a 04 00 00 00 00  P.........b*.....  0x42a6298(32@0x42a6290)
     0x42ae5b0 b8 6d 2a 04 00 00 00 00 00 00 00 00 00 00 00 00  .m*.............  0x42a6db8(32@0x42a6db0)
     0x42ae5c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
     0x42ae5d0 00 00 00 00 00 00 00 00 67 80 90 55 00 00 00 00  ........g..U....
     0x42ae5e0 40 00 00 00 00 00 00 00 08 e3 2a 04 00 00 00 00  @.........*.....  0x42ae308(112@0x42ae300)
     0x42ae5f0 28 ab 25 04 00 00 00 00 ba ba ba ab 65 00 74 00  (.%.........e.t.  0x425ab28(112@0x425ab20)
     0x42ae600 00 00 00 00 00 00 00 00 bf 97 b1 45 25 00 00 94  ...........E%...
                    Out<2> Plugin: show_allocation
```

MESSAGE_RECORD



Can we list all the messages?

```
[1] MirandaTest.E01 19:33:36> run -i /tmp/test.py
[1] MirandaTest.E01 19:36:26> a=miranda.MESSAGE_RECORD(0x42ae308)
[1] MirandaTest.E01 19:36:28> for x in a.walk_list("Next"):
                        |..>     print "%s %s: %s" % (x.Timestamp, x.Sender.deref(), x.Message.deref())
                        |..>
2015-06-28 23:16:46+0000 BeatrixKiddo: Thanks, I was worried for a second there.
2015-06-28 23:16:55+0000 Zereo: lol
2015-06-28 23:17:13+0000 fioco: I'll give free codes to all here with a steam that has been here before n
2015-06-28 23:17:24+0000 BeatrixKiddo: What?
2015-06-28 23:17:27+0000 fioco: So no to new comers that randomly join for a code :p
2015-06-28 23:17:29+0000 BeatrixKiddo: Was that even a sentence?
2015-06-28 23:17:39+0000 fioco: Yes
2015-06-28 23:17:43+0000 fioco: Yes it was
2015-06-28 23:18:08+0000 BeatrixKiddo: A sentence usually needs to be grammatically correct.
2015-06-28 23:18:28+0000 fioco: Minus maybe a comma or two, it was
2015-06-28 23:18:35+0000 Zereo: Ohh that reminds me I still got 2 castle crasher free game coupons from s
2015-06-28 23:18:37+0000 fioco: It's IRC not a business email ;p
2015-06-28 23:18:42+0000 fioco: I DONT
```

We can now write a plugin

- Extracting the internal Miranda state is as simple as understanding the data structures used by the application.
- No need to reverse engineer code in many cases.
  - It helps when we see the memory the way the application sees it:
    - Like size allocations have same functionality.
    - Can see interconnection between allocations.

Google | Rekall

Conclusions

- For the first time a FOSS memory analysis framework supports reliable user space address translation.
  - Prototype PTE, Page file, Transitioned PDEs etc.
  - High quality address translation is essential in order to reliably parse heap structures.
- Thorough heap analysis enables seeing memory through an app's own abstractions.

Future work.

- Have you ever been disappointed that *vaddump* or *dumpfiles* plugin produces files with missing pages?

```
0x0000738a0000         0x21000 File Mapping          \Windows\System32\msvcr100.dll (P)
0x0000738c1000          0x3000 Valid                 PhysAS @ 0x176ed000
0x0000738c4000          0x1000 File Mapping          \Windows\System32\msvcr100.dll @ 0x23400 (P)
0x0000738c5000          0x1000 Valid                 PhysAS @ 0x34d2e000
0x0000738c6000          0x6000 File Mapping          \Windows\System32\msvcr100.dll @ 0x25400 (P)
0x0000738cc000          0x3000 Valid                 PhysAS @ 0x1ac37000
0x0000738cf000          0x1000 File Mapping          \Windows\System32\msvcr100.dll @ 0x2e400 (P)
0x0000738d0000          0x1000 Valid                 PhysAS @ 0xe9e6000
0x0000738d1000          0xb000 File Mapping          \Windows\System32\msvcr100.dll @ 0x30400 (P)
0x0000738dc000          0x2000 Valid                 PhysAS @ 0x3d72000
0x0000738de000         0x15000 File Mapping          \Windows\System32\msvcr100.dll @ 0x3d400 (P)
0x0000738f3000          0x1000 Valid                 PhysAS @ 0x33eff000
0x0000738f4000         0x18000 File Mapping          \Windows\System32\msvcr100.dll @ 0x53400 (P)
0x00007390c000          0x2000 Valid                 PhysAS @ 0x25bf9000
0x00007390e000          0x2000 File Mapping          \Windows\System32\msvcr100.dll @ 0x6d400 (P)
0x000073910000          0x1000 Valid                 PhysAS @ 0x241000
0x000073911000          0x3000 File Mapping          \Windows\System32\msvcr100.dll @ 0x70400 (P)
0x000073914000          0x1000 Pagefile              PF 0 @ 0x323f9000
0x000073915000          0x1c000 File Mapping         \Windows\System32\msvcr100.dll @ 0x74400 (P)
0x000073931000          0x4000 Valid                 PhysAS @ 0x365f000
0x000073935000          0x1d000 File Mapping         \Windows\System32\msvcr100.dll @ 0x93c00 (P)
0x000073952000          0x2000 Valid                 PhysAS @ 0x20a37000
0x000073954000          0x3000 File Mapping          \Windows\System32\msvcr100.dll @ 0xb2800 (P)
0x000073957000          0x3000 Valid                 PhysAS @ 0x2b7d5000
0x00007395a000         0x18000 File Mapping          \Windows\System32\msvcr100.dll @ 0xb5a00 (P)
0x000077880000         0x21000 Valid                 PhysAS @ 0x1536000 (P)
0x0000778a1000          0x2000 Valid                 PhysAS @ 0x222ed000
0x0000778a3000          0x1000 Valid                 PhysAS @ 0x3fb6d000 (P)
0x0000778a4000          0x4000 Transition            PhysAS @ 0xde6e000 (P)
```

Google | Rekall

Future Work

- When you think about it - why do we ever dump files out of memory? Because we forgot to acquire them in the first place!
- Full system state = physical memory + pagefile + mapped files.
- We need better acquisition! Also grab mapped files!
- Coming soon to a Rekall near you!