The VAD Tree: A Process-Eye View of Physical Memory

By

Brendan Dolan-Gavitt

Presented At

The Digital Forensic Research Conference

DFRWS 2007 USA Pittsburgh, PA (Aug 13th - 15th)

The VAD Tree: A Process-Eye View of Physical Memory

Brendan Dolan-Gavitt

Too Much Data

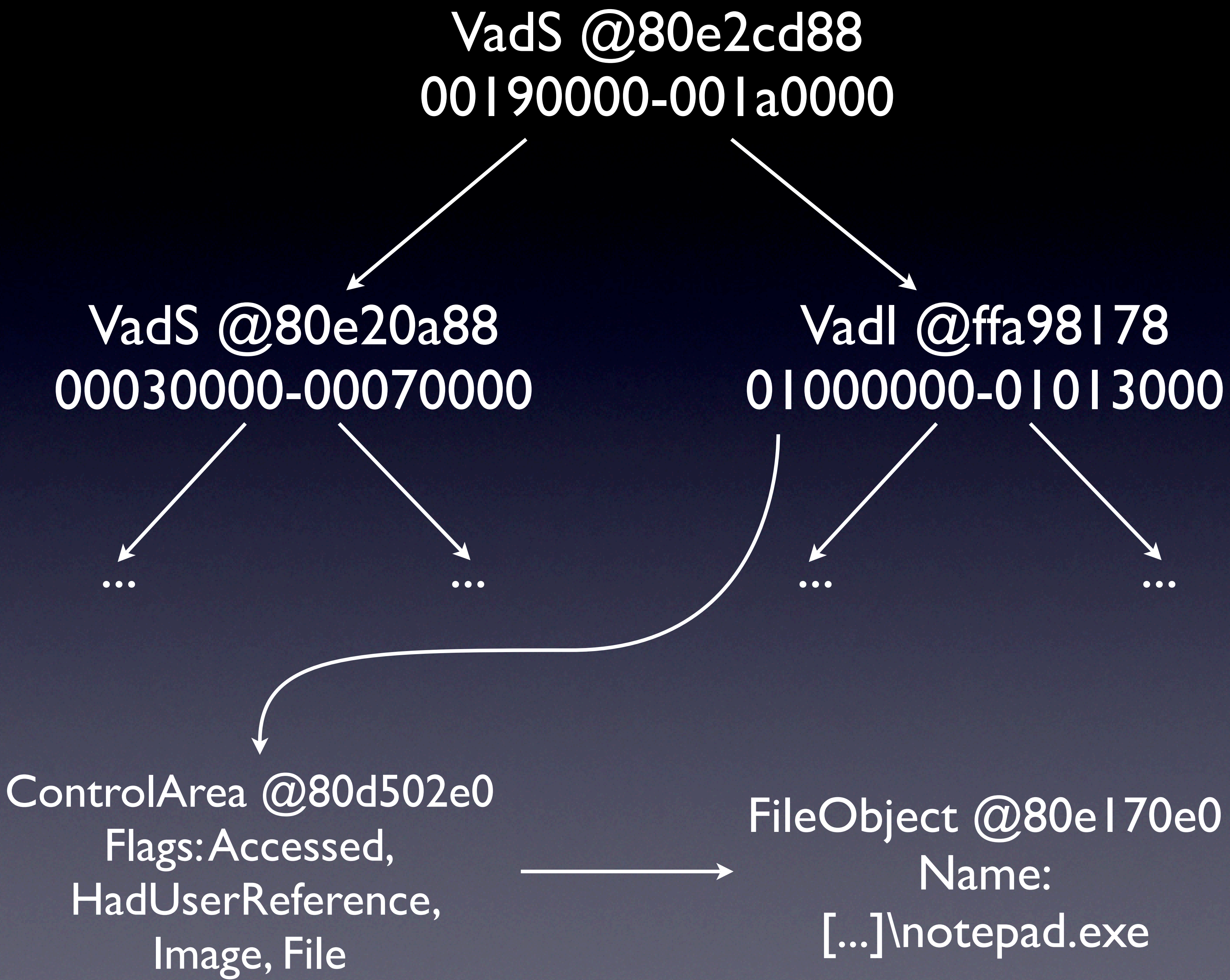
- Physical memory is voluminous, and getting more so
- Structure not well-understood for most operating systems (e.g. Windows)
- Want to give context to artifacts and evidence found in memory dumps

Current Approaches

- Attribution: look at the PDEs and PTEs for each process to determine what owned a particular bit of memory (e.g., Volatility's "strings" module)
- Find interesting structures such as processes, threads, kernel modules, TCP connections, etc. in memory

The VAD Tree

- Virtual Address Descriptor tree is a self-balancing binary tree that lists memory ranges allocated using VirtualAlloc().
- Three variations on the structure: short, normal, and long (`_MMVAD_SHORT`, `_MMVAD`, `_MMVAD_LONG` in XP SP2 debug symbols).


VadS @80e2cd88
00190000-001a0000
VadS @80e20a88
00030000-00070000
VadI @ffa98178
01000000-01013000
...
...
...
...
ControlArea @80d502e0
Flags: Accessed,
HadUserReference,
Image, File
FileObject @80e170e0
Name:
[...]\notepad.exe

Walking the VAD Tree

1. Find an `_EPROCESS` structure (using, e.g., Andreas Schuster's *PTFinder*).
2. Read the VadRoot member of that structure.
3. Use the pool tag (see Schuster IMF 2006) to determine the type of VAD node, then visit the left and right children.
4. GOTO 3.

Useful Properties

- VAD nodes correspond to the amount of memory requested by VirtualAlloc, rather than 4096 byte chunks as in page directory.
- Normal and long VADs contain pointers to Control Areas, which in turn point to File Objects if the memory range corresponds to a mapped file such as a DLL.
- Kernel memory structure–harder to modify.

VAD Tools

- vadwalk: walk the tree and give a short listing as ASCII art, table view, or GraphViz
- vadinfo: print detailed information on each node
- vaddump: write the memory ranges described by each node out to disk

Tool Limitations

- Naïve memory model (no support for “invalid” PDE/PTEs, such as prototype or transition pages).
- 32-bit, non-PAE mode is assumed.
- Only Windows 2000 and XP (SP0, SP1, SP2) supported–no Vista support.
- Code quality is “proof-of-concept”–expect crashes on strange inputs.

Anti-Forensics

- VAD Tree susceptible to DKOM attacks.
- Once the memory has been committed and the page directory entries created, VAD does not appear to be used.
- Code with access to kernel memory could remove a node from the tree without affecting the user-space process's ability to access it.

Future Work

- Vista VADs
- Finding and reconstructing VAD trees from exited processes (VadRoot member is zeroed out when process exits).
- Other interesting structures pointed to–for example, what are `_MMBANKED_SECTION` and `_MMEXTEND_INFO` ?

Links

- Volatility – GPL-licensed memory analysis framework by AAron Walters and Nick Petroni, Jr.
- VAD Tools – Public domain proof-of-concept tools.

Questions?