Surveying The User Space Through User Allocations

By

Andrew White, Bradley Schatz and Ernest Foo

Presented At

The Digital Forensic Research Conference

DFRWS 2012 USA Washington, DC (Aug 6th - 8th)

Surveying The User Space Through User Allocations

Andrew White, Bradley Schatz, Ernest Foo
Queensland University of Technology
Brisbane Australia

DFRWS

6th August 2012

Contributions

- Framework for describing userspace memory
- Analysis of metadata sources for userspace memory
- Implementation of analysis and framework as a Volatility plugin

Cavet

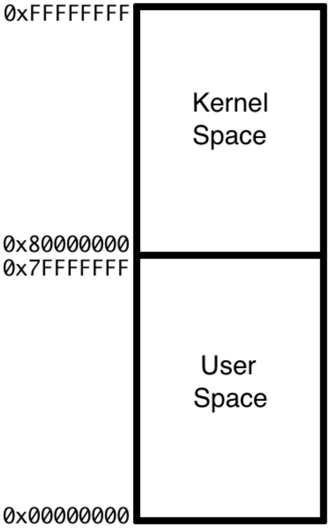
- All memory related information in this talk relates to 32-bit Windows in default configurations

Physical and Virtual Memory

- Physical memory
  - The amount of physical RAM a machine has available, eg 2GB
- Virtual memory
  - The memory address space presented to each process by the system
  - 4GB on 32-bit Windows
- Provides numerous advantages to the system, eg isolation of kernel and user space code

Kernel and User Space

- Kernel Space
  - Where code and data required by the operating system is stored
    - E.g. Processes, Sockets, Drivers
- User Space
  - Where code and data required by a process is stored
    - E.g. Threads, EXE/DLLs, Heaps



Problem

- Memory forensic research has focused on:
  - Kernel space memory
  - Recreation of live response tools
  - Locating specific data structures
- While all valuable contributions, research has mostly ignored:
  - User space memory
  - Improving how memory is explored

Motivation

- Difficult to find specific information used by a process
  - E.g. encryption keys, passwords
  - Currently would have to reverse the program or search memory with strings
- Identifying user space malware
  - Current artefact-based techniques will only last so long
  - Facilitate the development of new techniques

Related Work

- Memparser (Betz, 2005) - PEB, Loaded modules, Process Parameters
- Dolan-Gavitt (2007) - VAD Tree, File Objects
- Arasth and Debbabi (2007) - TEB, Stacks

User Allocations

- Windows memory manager tracks user space memory allocations using Virtual Address Descriptors (VADs)
- Each process has its own tree structure of VADs to store information about the memory being allocated
  - The address range, permissions, etc.
- We term each of the allocated memory ranges that a VAD describes a *user allocation*
- We use these user allocations as a framework within which to explore user space memory

Assumptions

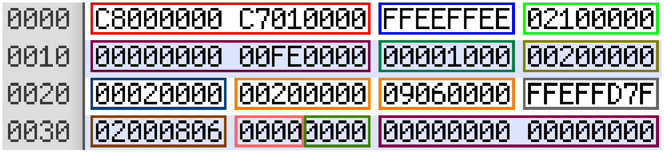
- Does the VAD Tree really describe *all* user allocations?
- The answer - not quite
  - Every process on WinXP SP3 and Win7 SP1 has one user allocation at 0x7FFE0000 not described by the VAD Tree
  - Each of these user allocations map to the same physical page
  - Contains the _KUSER_SHARED_DATA structure
- Possible that malicious code could use memory without a VAD in the same way
  - Such allocations need to be checked for when using the VAD Tree

Data versus Metadata

- Each user allocation contains data of some sort

```
0000 | C8000000 C7010000 FFEEFFEE 02100000
0010 | 00000000 00FE0000 00001000 00200000
0020 | 00020000 00200000 09060000 FFEFFD7F
0030 | 02000806 00000000 00000000 00000000
```

- In order to understand that data, we need metadata about what that data represents



```
0000 | C8000000 C7010000 FFEEFFEE 02100000
0010 | 00000000 00FE0000 00001000 00200000
0020 | 00020000 00200000 09060000 FFEFFD7F
0030 | 02000806 00000000 00000000 00000000
```

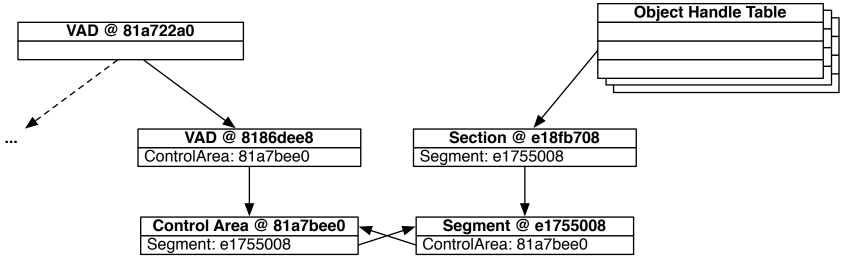
Metadata Types

- Structural
  - Describes the structure the data is stored in
  - Allows us to find specific objects
- Descriptive
  - Describes what the data represents
  - Allows us to determine the purpose of the user allocation
- While ideally we want both, sometimes we can only find one or the other
- The metadata we find can be categorized based on its origin
  - Kernel or user space

Kernel Space Metadata Sources

- VADs have two types
  - Private - Exists only in this address space
  - Mapped - Exists in multiple address spaces
  - Indicated by whether VAD references a ControlArea
- File Objects
  - Memory mapped files
- Sections
  - Represents shared memory
  - Indicates that processes are communicating in some way
  - Can be found by parsing the object manager

Mapping a Section to a VAD



User Space Metadata Sources

- Process Environment Block (PEB)
  - Allows the location of numerous data structures
  - Heaps, Code Pages, Activation Contexts, Shim Data, etc.
- Thread Environment Block (TEB)
  - Stacks, Threads
- Heaps
  - Some heap structures occupy their own user allocation
  - Heap Segments, Heap Virtual Allocations

Using Metadata

- Link metadata to its relevant user allocation
    - Reveals content or purpose of the allocation
- The metadata allows us to narrow the search range for data of interest
    - E.g. user data
        - Search only data storage user allocations, eg heaps, stacks, files
        - Can safely ignore code and default process user allocations, e.g. EXE/DLLs, activation contexts, code pages

Implementation

- Implemented analysis of metadata sources as a Volatility plugin
- Userspace.py
  - Links user allocations with related metadata
  - Allows content or purpose of user allocation to be known
  - Tested on WinXP SP3 x86 and Win7 SP1 x86
- Created two versions for evaluation purposes
  - Existing.py - Only metadata previous research could identify
  - Userspace.py - Metadata our research additionally identifies

malloc.exe on WinXP SP3 - Existing.py

```
Start    End      Used     Size     Permission        Type     Description
-------- -------- -------- -------- ----------------- -------- ----------------------------------------------
00010000 00010fff 00001000 00001000 READWRITE         Private
00020000 00020fff 00001000 00001000 READWRITE         Private  Parameters
00030000 0012ffff 00002000 00100000 READWRITE         Private  Stack (Thread 1)
00130000 00132fff 00002000 00003000 READONLY          Mapped
00140000 0023ffff 00004000 00100000 READWRITE         Private
00240000 0024ffff 00003000 00010000 READWRITE         Private
00250000 0025ffff 00002000 00010000 READWRITE         Mapped
00260000 00275fff 00005000 00016000 READONLY          Mapped   \WINDOWS\system32\unicode.nls
00280000 002c0fff 00001000 00041000 READONLY          Mapped   \WINDOWS\system32\locale.nls
002d0000 00310fff 00000000 00041000 READONLY          Mapped   \WINDOWS\system32\sortkey.nls
00320000 00325fff 00004000 00006000 READONLY          Mapped   \WINDOWS\system32\sorttbls.nls
00330000 0033ffff 00005000 00010000 READWRITE         Private
00340000 00342fff 00002000 00003000 READONLY          Mapped   \WINDOWS\system32\ctype.nls
00350000 003d0fff 00081000 00081000 READWRITE         Private
00400000 0040efff 0000e000 0000f000 EXECUTE_WRITECOPY Mapped   \Documents and Settings\xp\malloc.exe
7c800000 7c8f5fff 0002b000 000f6000 EXECUTE_WRITECOPY Mapped   \WINDOWS\system32\kernel32.dll
7c900000 7c9b1fff 0002f000 000b2000 EXECUTE_WRITECOPY Mapped   \WINDOWS\system32\ntdll.dll
7f6f0000 7f7effff 00002000 00100000 EXECUTE_READ      Mapped
7ffb0000 7ffd3fff 00005000 00024000 READONLY          Mapped
7ffd7000 7ffd7fff 00001000 00001000 READWRITE         Private  PEB
7ffdf000 7ffdffff 00001000 00001000 READWRITE         Private  TEB (Thread 1)
```

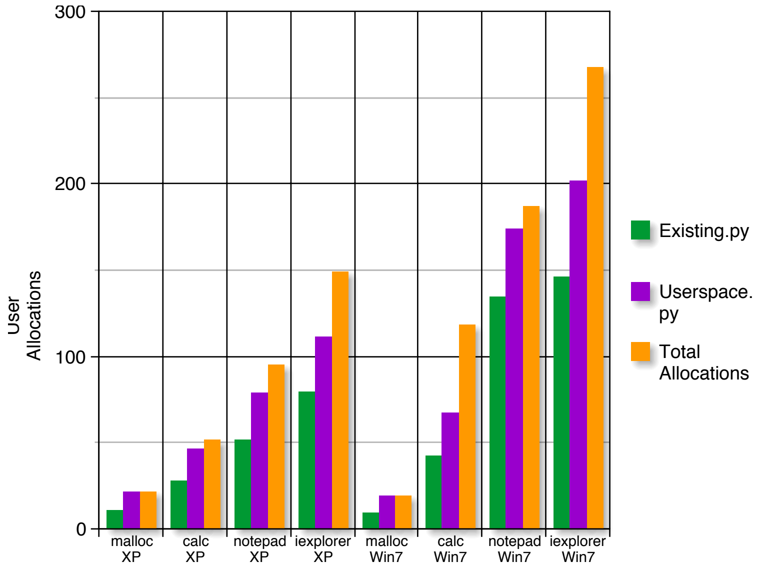
malloc.exe on WinXP SP3 - Userspace.py

```
Start    End      Used     Size     Permission        Type     Description
-------- -------- -------- -------- ----------------- -------- -------------------------------------------
00010000 00010fff 00001000 00001000 READWRITE         Private  Environment
00020000 00020fff 00001000 00001000 READWRITE         Private  Parameters
00030000 0012ffff 00002000 00100000 READWRITE         Private  Stack (Thread 1)
00130000 00132fff 00002000 00003000 READONLY          Mapped   SystemDefaultActivationContextData
                                                               Section - PID 00584, Name ‘’
00140000 0023ffff 00004000 00100000 READWRITE         Private  Heap 0
00240000 0024ffff 00003000 00010000 READWRITE         Private  Heap 1
00250000 0025ffff 00002000 00010000 READWRITE         Mapped   Heap 2
00260000 00275fff 00005000 00016000 READONLY          Mapped   \WINDOWS\system32\unicode.nls
                                                               Section - PID 00584, Name ‘NlsSectionUnicode’
00280000 002c0fff 00001000 00041000 READONLY          Mapped   \WINDOWS\system32\locale.nls
                                                               Section - PID 00584, Name ‘NlsSectionLocale’
002d0000 00310fff 00000000 00041000 READONLY          Mapped   \WINDOWS\system32\sortkey.nls
                                                               Section - PID 00584, Name ‘NlsSectionSortkey’
00320000 00325fff 00004000 00006000 READONLY          Mapped   \WINDOWS\system32\sorttbls.nls
                                                               Section - PID 00584, Name ‘NlsSectionSortTbls’
00330000 0033ffff 00005000 00010000 READWRITE         Private  Heap 3
00340000 00342fff 00002000 00003000 READONLY          Mapped   \WINDOWS\system32\ctype.nls
                                                               Section - PID 00584, Name ‘NlsSectionCType’
00350000 003d0fff 00081000 00081000 READWRITE         Private  Virtual Alloc of Heap 3
00400000 0040efff 0000e000 0000f000 EXECUTE_WRITECOPY Mapped   \Documents and Settings\xp\malloc.exe
7c800000 7c8f5fff 0002b000 000f6000 EXECUTE_WRITECOPY Mapped   \WINDOWS\system32\kernel32.dll
7c900000 7c9b1fff 0002f000 000b2000 EXECUTE_WRITECOPY Mapped   \WINDOWS\system32\ntdll.dll
7f6f0000 7f7effff 00002000 00100000 EXECUTE_READ      Mapped   Shared Heap
                                                               Section - PID 00584, Name ‘’
7ffb0000 7ffd3fff 00005000 00024000 READONLY          Mapped   Code Page
7ffd7000 7ffd7fff 00001000 00001000 READWRITE         Private  PEB
7ffdf000 7ffdffff 00001000 00001000 READWRITE         Private  TEB (Thread 1)
7ffe0000 7ffe0fff 00001000 00001000 N/A               N/A      User Shared Data
```

Plugin Comparison

- Compared results of existing.py and userspace.py on a number of processes
  - malloc.exe - extremely simple C program that allocates memory
  - calc.exe, notepad.exe, iexplorer.exe - standard Windows applications
  - For Windows programs, default OS versions were used
  - All programs run through basic usage scenarios
- Averaged results across multiple runs on different VMs with different memory configurations
- Compared to average number of user allocations from each process
  - Determined from analysis of VAD Tree and completeness scan

Allocations explained by existing.py and userspace.py



Limitations

- In general, the more complex the application, the lower the percentage of user allocations that were explained
- Two possible explanations
  - Metadata for unexplained allocations does not exist
  - Metadata for unexplained allocations exist in an unexaminated source of metadata

Conclusion

- Framework for analysis of user space memory
- Analysis of user space memory metadata sources
- Implementation as a Volatilty plugin
- Facilitate future analysis of user space memory

Future Work

- Increase the granularity of analysis
    - Physical -> Virtual -> Allocations -> Objects
    - Object identification
    - Use new information to improve malware detection
- Resistance of metadata to malicious modification
- 64-bit / Windows 8 support

Questions?

- Code available at `http://github.com/a-white/`
- Slides available at `http://tinyurl.com/dfrws-a-white`
- Questions?

Bibliography

- Arasteh AR, Debbabi M. Forensic memory analysis: From stack and code to execution history. Digital Investigation 2007;4(S1):114-25.
- Betz C. MemParser. `http://www.dfrws.org/2005/challenge/memparser.shtml`; 2005.
- Dolan-Gavitt B. The VAD tree: A process-eye view of physical memory. Digital Investigation 2007;4(S1):62-4.

More sample output [1/3]

```
Start    End      Used     Size     Permission        Type     Description
-------- -------- -------- -------- ----------------- -------- ------------------------------------------------
00010000 0001ffff 00001000 00010000 READWRITE         Mapped   Heap 1
00020000 00022fff 00003000 00003000 WRITECOPY         Mapped   \Windows\System32\en-US\notepad.exe.mui
00030000 00033fff 00004000 00004000 READONLY          Mapped   SystemDefaultActivationContextData
                                                               Section - PID 00356, Name ‘’
00040000 00041fff 00002000 00002000 READONLY          Mapped   ActivationContextData
                                                               Section - PID 00356, Name ‘’
00050000 00050fff 00001000 00001000 READWRITE         Private  pContextData
00060000 00060fff 00001000 00001000 READWRITE         Private  (GDI Data)
00070000 00070fff 00001000 00001000 READWRITE         Private  (GDI Data)
00080000 00080fff 00001000 00001000 READWRITE         Mapped   Section - PID 00308, Name ‘windows_shell_global_
00090000 00091fff 00002000 00002000 READONLY          Mapped   Section - PID 00356, Name ‘’
000c0000 001bffff 00027000 00100000 READWRITE         Private  Heap 0 (GDI Data)
001d0000 0020ffff 00003000 00040000 READWRITE         Private  Stack of Thread 0
00210000 00276fff 00030000 00067000 READONLY          Mapped   \Windows\System32\locale.nls
00280000 00347fff 00005000 000c8000 READONLY          Mapped
00390000 0039ffff 00001000 00010000 READWRITE         Private  Heap 3
003a0000 0041ffff 00001000 00080000 READWRITE         Private
00440000 0044ffff 00003000 00010000 READWRITE         Private  Heap 2
00450000 00550fff 0000b000 00101000 READONLY          Private  GdiSharedHandleTable
00560000 0063efff 00053000 000df000 READONLY          Mapped   Section - PID 00308, Name ‘’
00700000 0073ffff 00010000 00040000 READWRITE         Private  Heap 4
00830000 0086ffff 00014000 00040000 READWRITE         Private  Heap 5
00870000 00b3efff 0001f000 002cf000 READONLY          Mapped   \Windows\Globalization\Sorting\SortDefault.nls
00c00000 00c2ffff 0000f000 00030000 EXECUTE_WRITECOPY Mapped   \Windows\System32\notepad.exe
00c30000 0182ffff 0000e000 00c00000 READONLY          Mapped
01830000 0215ffff 0003e000 00930000 READONLY          Mapped   \Windows\Fonts\StaticCache.dat
                                                               Section - PID 00308, Name ‘’
```

More sample output [2/3]

```
Start    End       Used     Size      Permission        Type     Description
-------- --------  -------- --------  ----------------- -------- ---------------------------------------------
72e00000 72e50fff  00013000 00051000  EXECUTE_WRITECOPY Mapped   \Windows\System32\winspool.drv
73a00000 73a12fff  00009000 00013000  EXECUTE_WRITECOPY Mapped   \Windows\System32\dwmapi.dll
73d30000 73d6ffff  0001c000 00040000  EXECUTE_WRITECOPY Mapped   \Windows\System32\uxtheme.dll
73eb0000 7404dfff  0003c000 0019e000  EXECUTE_WRITECOPY Mapped   \Windows\winsxs\x86_microsoft.windows.common-[.
74420000 74428fff  00007000 00009000  EXECUTE_WRITECOPY Mapped   \Windows\System32\version.dll
74e70000 74e7bfff  00008000 0000c000  EXECUTE_WRITECOPY Mapped   \Windows\System32\cryptbase.dll
74fd0000 75019fff  0001d000 0004a000  EXECUTE_WRITECOPY Mapped   \Windows\System32\KernelBase.dll
75220000 752bcfff  00043000 0009d000  EXECUTE_WRITECOPY Mapped   \Windows\System32\usp10.dll
752c0000 7536bfff  0001c000 000ac000  EXECUTE_WRITECOPY Mapped   \Windows\System32\msvcrt.dll
75370000 754cbfff  00019000 0015c000  EXECUTE_WRITECOPY Mapped   \Windows\System32\ole32.dll
754d0000 7551dfff  00015000 0004e000  EXECUTE_WRITECOPY Mapped   \Windows\System32\gdi32.dll
75520000 75576fff  00010000 00057000  EXECUTE_WRITECOPY Mapped   \Windows\System32\shlwapi.dll
758f0000 75990fff  00013000 000a1000  EXECUTE_WRITECOPY Mapped   \Windows\System32\rpcrt4.dll
75a00000 75ad3fff  00029000 000d4000  EXECUTE_WRITECOPY Mapped   \Windows\System32\kernel32.dll
75ae0000 75b6efff  0000b000 0008f000  EXECUTE_WRITECOPY Mapped   \Windows\System32\oleaut32.dll
75b70000 767b9fff  00017000 00c4a000  EXECUTE_WRITECOPY Mapped   \Windows\System32\shell32.dll
767c0000 76888fff  00028000 000c9000  EXECUTE_WRITECOPY Mapped   \Windows\System32\user32.dll
76a90000 76b0afff  00008000 0007b000  EXECUTE_WRITECOPY Mapped   \Windows\System32\comdlg32.dll
76cb0000 76d4ffff  00011000 000a0000  EXECUTE_WRITECOPY Mapped   \Windows\System32\advapi32.dll
76d50000 76d59fff  00007000 0000a000  EXECUTE_WRITECOPY Mapped   \Windows\System32\lpk.dll
76d60000 76d7efff  0000a000 0001f000  EXECUTE_WRITECOPY Mapped   \Windows\System32\imm32.dll
76dd0000 76f0bfff  0005d000 0013c000  EXECUTE_WRITECOPY Mapped   \Windows\System32\ntdll.dll
76f10000 76f28fff  00008000 00019000  EXECUTE_WRITECOPY Mapped   \Windows\System32\sechost.dll
76f30000 76ffbfff  00025000 000cc000  EXECUTE_WRITECOPY Mapped   \Windows\System32\msctf.dll
77010000 77010fff  00001000 00001000  EXECUTE_WRITECOPY Mapped   \Windows\System32\apisetschema.dll
```

More sample output [3/3]

```
Start    End       Used     Size      Permission        Type     Description
-------- --------  -------- --------  ----------------- -------- ----------------------------------------------
7f6f0000 7f7effff  00005000 00100000  READONLY          Mapped   Heap 6 (Shared)
                                                                 Section - PID 00356, Name ‘SharedSection’
7ffb0000 7ffd2fff  00013000 00023000  READONLY          Mapped   Code Page
7ffd4000 7ffd4fff  00001000 00001000  READWRITE         Private  PEB
7ffdf000 7ffdffff  00001000 00001000  READWRITE         Private  TEB (Thread 0)
7ffe0000 7ffe0fff  00001000 00001000  N/A               N/A      KSHARED_USER_DATA

Unreferenced Pages
Start    Size
```

Figure: *Userspace.py* running against *notepad.exe* on Windows 7.