Heap Spray Detection with Heap Inspector

Aaron LeMasters
MANDIANT

Please complete the Speaker Feedback Surveys!

Introduction

- About me
- Purpose of this talk
- Goals
  - What is an application storing? How is it storing it?
  - Visualization
- Current research
  - EMET, STRIDE, Nozzle, HeapLocker

What can I do with this tool?

- We will focus on two primary use cases:
  - Detect/visualize heap sprays
  - Search for PII
- Other uses
  - Reverse memory structures
  - Debug heap anomalies
  - Vulnerability research / exploit dev (future)

Features

- View heap allocations in a spatial arrangement
- View heap contents in an embedded hex viewer
- Search for byte patterns, regexes and strings
- Export heap chunks to use in other tools
- It comes in two forms:
  - Command line exe/dll
  - C# user interface

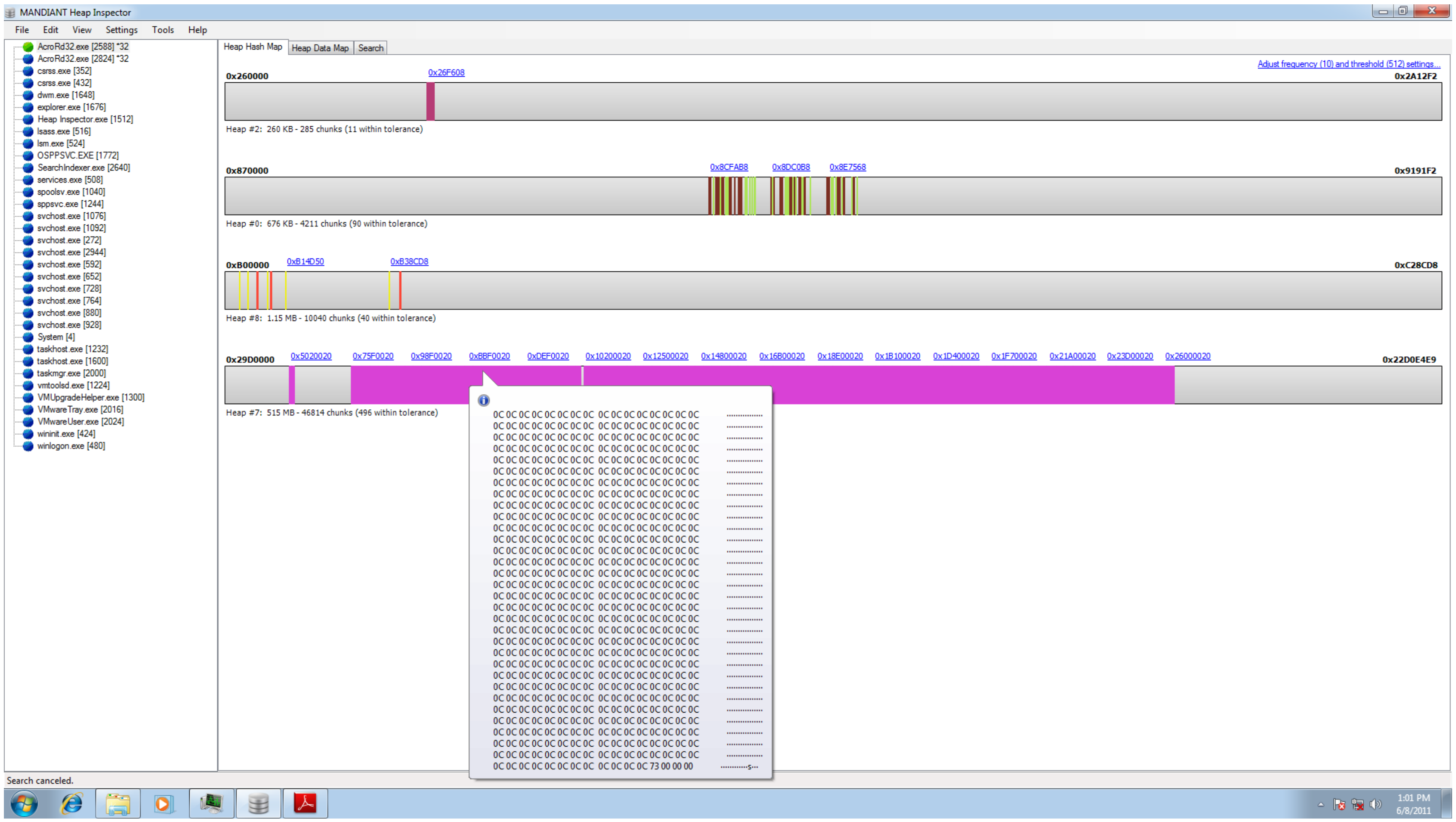
Basic Windows Heap Mechanics

- A process has a default heap and one or more private heaps:
  - Heaps are made up of one or more segments
  - Segments are made up of one or more chunks
  - Chunks have the data you care about
- This is all you need to know to understand Heap Inspector
  - For an in-depth discussion of heap internals, see Chris Valacek's talk

How does a heap spray work?

- A heap spray is a technique to stage shellcode
- Meant to increase the reliability of exploiting memory corruption vulnerability
- Most commonly seen in applications that host JIT engines (flash, java, etc), such as web browsers and document readers (Adobe, MS Word)
  - CVE-2011-0609, CVE-2010-1297, CVE-2010-3973, CVE-2010-3971, just to name a few
- Heap spraying just allocates the same block of data hundreds of times
  - We use this to our advantage

Heap Inspector User Interface


MANDIANT Heap Inspector

File Edit View Settings Tools Help

- AcroRd32.exe [2588] *32
- AcroRd32.exe [2824] *32
- csrss.exe [352]
- csrss.exe [432]
- dwm.exe [1648]
- explorer.exe [1676]
- Heap Inspector.exe [1512]
- lsass.exe [516]
- lsm.exe [524]
- OSPPSVC.EXE [1772]
- SearchIndexer.exe [2640]
- services.exe [508]
- spoolsv.exe [1040]
- sppsvc.exe [1244]
- svchost.exe [1076]
- svchost.exe [1092]
- svchost.exe [272]
- svchost.exe [2944]
- svchost.exe [592]
- svchost.exe [652]
- svchost.exe [728]
- svchost.exe [764]
- svchost.exe [880]
- svchost.exe [928]
- System [4]
- taskhost.exe [1232]
- taskhost.exe [1600]
- taskmgr.exe [2000]
- vmtoolsd.exe [1224]
- VMUpgradeHelper.exe [1300]
- VMwareTray.exe [2016]
- VMwareUser.exe [2024]
- wininit.exe [424]
- winlogon.exe [480]

Heap Hash Map | Heap Data Map | Search

Adjust frequency (10) and threshold (512) settings...

0x260000 — 0x26F608 — 0x2A12F2

Heap #2: 260 KB - 285 chunks (11 within tolerance)

0x870000 — 0x8CFAB8 — 0x8DC0B8 — 0x8E7568 — 0x9191F2

Heap #0: 676 KB - 4211 chunks (90 within tolerance)

0xB00000 — 0xB14D50 — 0xB38CD8 — 0xC28CD8

Heap #8: 1.15 MB - 10040 chunks (40 within tolerance)

0x29D0000 — 0x5020020 — 0x75F0020 — 0x98F0020 — 0xBBF0020 — 0xDEF0020 — 0x10200020 — 0x12500020 — 0x14800020 — 0x16B00020 — 0x18E00020 — 0x1B100020 — 0x1D400020 — 0x1F700020 — 0x21A00020 — 0x23D00020 — 0x26000020 — 0x22D0E4E9

Heap #7: 515 MB - 46814 chunks (496 within tolerance)

```
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C    ................
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 73 00 00 00    ............s...
```

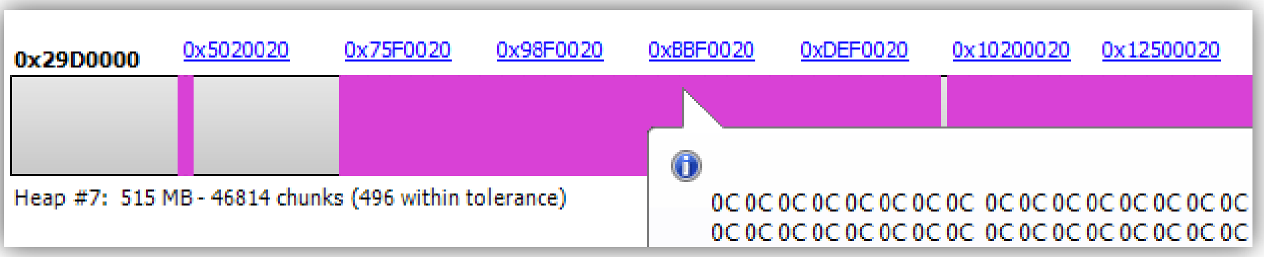
Search canceled.

1:01 PM
6/8/2011

The heap hash map

Looking for heap sprays

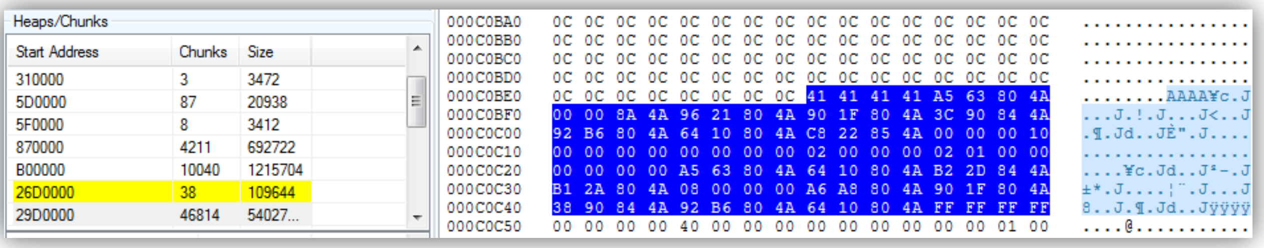
- Groups heap chunks across all heaps that have the same CRC32 (same color = same hash)
- Useful for spotting heap sprays



Heap #7: 515 MB - 46814 chunks (496 within tolerance)

```
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C
0C 0C 0C 0C 0C 0C 0C 0C  0C 0C 0C 0C 0C 0C 0C 0C
```

Visualization of successful heap spray in Adobe Reader (CVE-2010-2883)



Heaps/Chunks

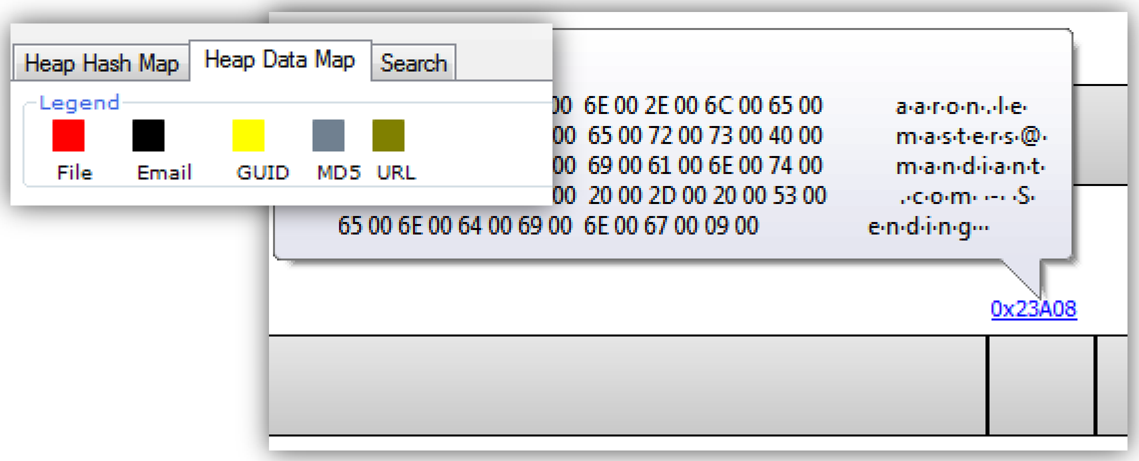
Start Address	Chunks	Size
310000	3	3472
5D0000	87	20938
5F0000	8	3412
870000	4211	692722
B00000	10040	1215704
26D0000	38	109644
29D0000	46814	54027...

```
000C0BA0   0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C   ................
000C0BB0   0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C   ................
000C0BC0   0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C   ................
000C0BD0   0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C   ................
000C0BE0   0C 0C 0C 0C 0C 0C 0C 0C 41 41 41 41 A5 63 80 4A   ........AAAA¥c.J
000C0BF0   00 00 8A 4A 96 21 80 4A 90 1F 80 4A 3C 90 84 4A   ...J.!.J...J<..J
000C0C00   92 B6 80 4A 64 10 80 4A C8 22 85 4A 00 00 00 10   .¶.Jd..JÈ".J....
000C0C10   00 00 00 00 00 00 00 00 02 00 00 00 02 01 00 00   ................
000C0C20   00 00 00 00 A5 63 80 4A 64 10 80 4A B2 2D 84 4A   ....¥c.Jd..J²-.J
000C0C30   B1 2A 80 4A 08 00 00 00 A6 A8 80 4A 90 1F 80 4A   ±*.J....¦¨.J...J
000C0C40   38 90 84 4A 92 B6 80 4A 64 10 80 4A FF FF FF FF   8..J.¶.Jd..Jÿÿÿÿ
000C0C50   00 00 00 00 40 00 00 00 00 00 00 00 00 00 01 00   ....@...........
```

The heap data map

Looking for PII

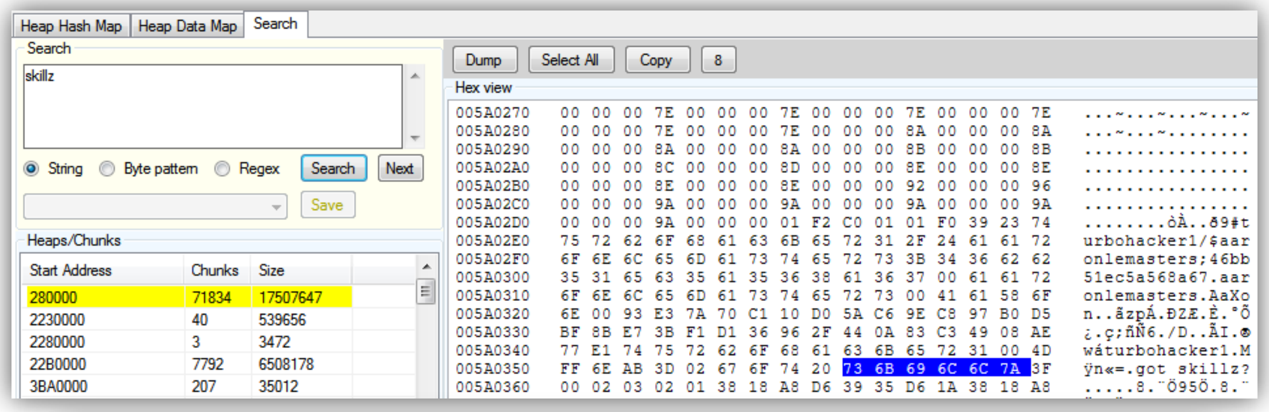
- Overlay regular expression matches on the heap map



```
00 6E 00 2E 00 6C 00 65 00       a·a·r·o·n·.·l·e·
00 65 00 72 00 73 00 40 00       m·a·s·t·e·r·s·@·
00 69 00 61 00 6E 00 74 00       m·a·n·d·i·a·n·t·
00 20 00 2D 00 20 00 53 00       .·c·o·m· ·-· ·S·
65 00 6E 00 64 00 69 00 6E 00 67 00 09 00       e·n·d·i·n·g···
```

Searching

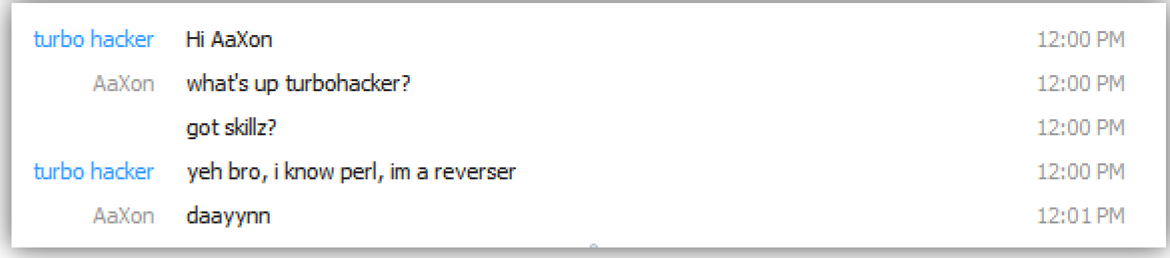
- String (unicode/ascii), byte and regex searching



skillz

Start Address	Chunks	Size
280000	71834	17507647
2230000	40	539656
2280000	3	3472
22B0000	7792	6508178
3BA0000	207	35012

```
005A0270   00 00 00 7E 00 00 00 7E 00 00 00 7E 00 00 00 7E   ...~...~...~...~
005A0280   00 00 00 7E 00 00 00 7E 00 00 00 8A 00 00 00 8A   ...~...~........
005A0290   00 00 00 8A 00 00 00 8A 00 00 00 8B 00 00 00 8B   ................
005A02A0   00 00 00 8C 00 00 00 8D 00 00 00 8E 00 00 00 8E   ................
005A02B0   00 00 00 8E 00 00 00 8E 00 00 00 92 00 00 00 96   ................
005A02C0   00 00 00 9A 00 00 00 9A 00 00 00 9A 00 00 00 9A   ................
005A02D0   00 00 00 9A 00 00 00 01 F2 C0 01 01 F0 39 23 74   ........òÀ..ð9#t
005A02E0   75 72 62 6F 68 61 63 6B 65 72 31 2F 24 61 61 72   urbohacker1/$aar
005A02F0   6F 6E 6C 65 6D 61 73 74 65 72 73 3B 34 36 62 62   onlemasters;46bb
005A0300   35 31 65 63 35 61 35 36 38 61 36 37 00 61 61 72   51ec5a568a67.aar
005A0310   6F 6E 6C 65 6D 61 73 74 65 72 73 00 41 61 58 6F   onlemasters.AaXo
005A0320   6E 00 93 E3 7A 70 C1 10 D0 5A C6 9E C8 97 B0 D5   n..ãzpÁ.ÐZÆ.È.°Õ
005A0330   BF 8B E7 3B F1 D1 36 96 2F 44 0A 83 C3 49 08 AE   ¿.ç;ñÑ6./D..ÃI.®
005A0340   77 E1 74 75 72 62 6F 68 61 63 6B 65 72 31 00 4D   wáturbohacker1.M
005A0350   FF 6E AB 3D 02 67 6F 74 20 73 6B 69 6C 6C 7A 3F   ÿn«=.got skillz?
005A0360   00 02 03 02 01 38 18 A8 D6 39 35 D6 1A 38 18 A8   .....8.¨Ö95Ö.8.¨
```



turbo hacker	Hi AaXon	12:00 PM
AaXon	what's up turbohacker?	12:00 PM
	got skillz?	12:00 PM
turbo hacker	yeh bro, i know perl, im a reverser	12:00 PM
AaXon	daayynn	12:01 PM

- This looks like some sort of data structure…

How does it work?

- C# application injects a C++ DLL using standard DLL injection
  - Also supported: LdrLoadDll and Reflective Injection [3]
- DLL acts as a server, receives messages from C# app and sends back data over named pipe
  - C#/Interop
  - Uses standard Win32 heap walking API’s
  - Raw parsing partially implemented
- Why Inject?
  - To get access to private heaps!

Caveats and Technical challenges

- DLL injection inherent caveats:
  - Instability due to synchronization issues (single-threaded to multi-threaded – thread safe?)
  - Instability due to deadlock conditions: accessing/locking heaps in use
  - Upon loading, entry point of every other DLL in process is called (side effects??)
- Sandboxed processes (ahem, Chrome):
  - Hooking
  - Least-privilege, isolation (job object, different desktop)
  - Injection solution: Use Stephen Fewer’s reflective DLL injection technique
    - Problem: least privilege token –can’t do anything!
- Other issues
  - Injecting into a service
  - Session separation introduced in Vista
    - Use NtCreateThreadEx
  - Universal injection across sessions
    - Terminal services (XP), Vista session separation
  - Wow64/Stub32
  - Access violations: use SEH instead of C++ exception handling
  - Smss.exe – doesn’t fully map in kernel32.dll – AV = BSOD!

DEMO: Extracting shellcode from a successful heap spray

But I can already do that …

- Debugger
  - Requires skillz – OS/heap internals knowledge
- Instrumentation
  - Requires code analysis, disassembly, heuristics
  - Overhead
  - False +/-
- Memory analysis
  - Requires OS internals knowledge
  - Data explosion
  - Smear
  - Stale

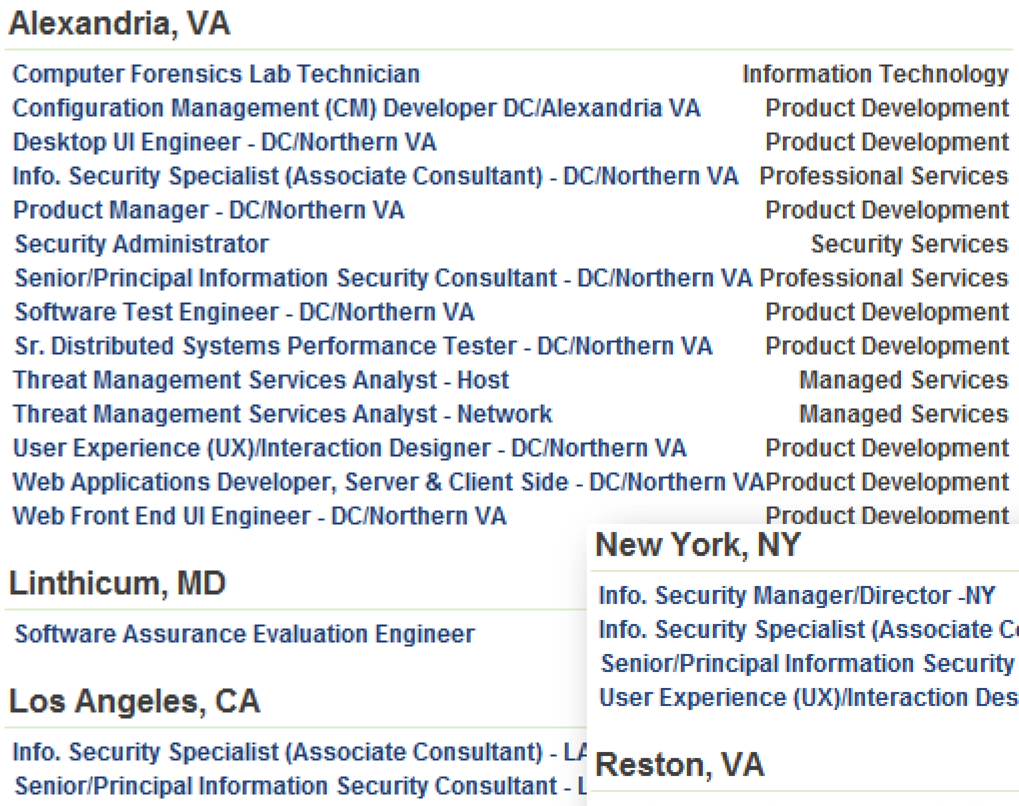
Future Direction

- Real-time detection of heap-spray
- Vulnerability research applications
  - Real-time heap modification
  - Taint analysis through “heap stalking”
- Memory images as input
  - Will take advantage of raw method

References / Further reading

- Chris Valasek, *Understanding the Low Fragmentation Heap*, http://illmatics.com/Understanding_the_LFH.pdf
- Microsoft, *CreateRemoteThread*, http://msdn.microsoft.com/en-us/library/ms682437%28VS.85%29.aspx
- Stephen Fewer, Reflective DLL Injection, http://www.harmonysecurity.com/files/HS-P005_ReflectiveDllInjection.pdf
- Didier Stevens, *HeapLocker*, http://blog.didierstevens.com/2010/12/06/heaplocker/
- Microsoft Research, *Nozzle*, http://research.microsoft.com/apps/pubs/default.aspx?id=76528
- Microsoft Research, *Enhanced Mitigation Experience Toolkit v2.0*, http://www.microsoft.com/download/en/details.aspx?id=5419
- Akritidis, et al, *STRIDE: Polymorphic Sled Detection Through Instruction Sequence Analysis*, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.79.5094&rep=rep1&type=pdf
- Bania, *JIT Spraying and Mitigations*, http://www.kryptoslogic.com/download/JIT_Mitigations.pdf

MANDIANT is hiring!



Alexandria, VA

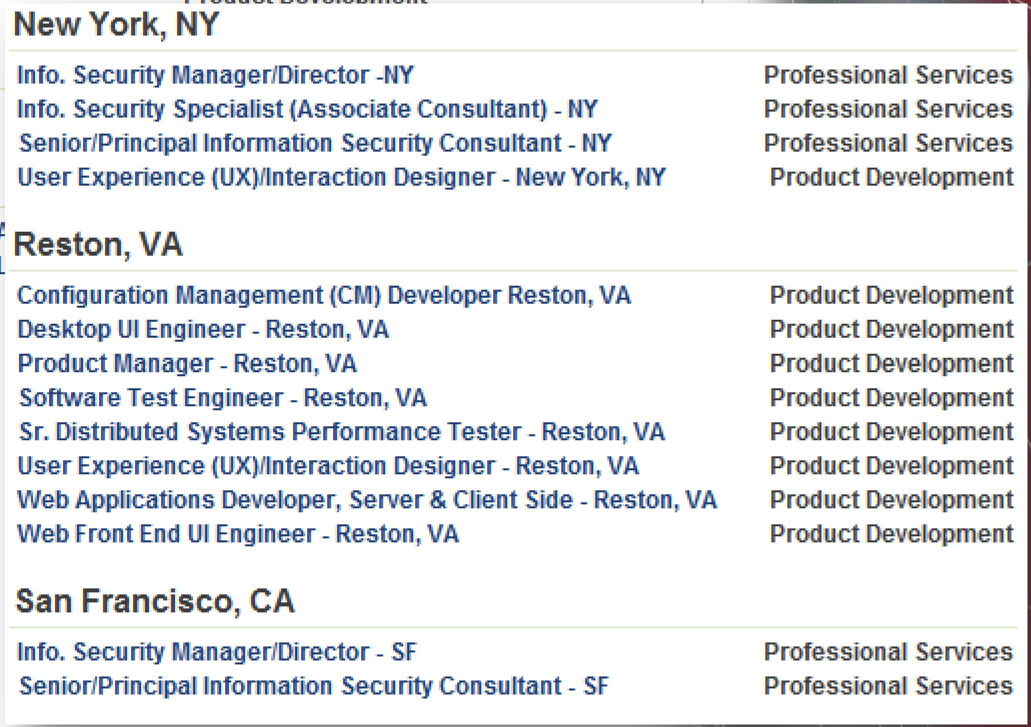
Computer Forensics Lab Technician	Information Technology
Configuration Management (CM) Developer DC/Alexandria VA	Product Development
Desktop UI Engineer - DC/Northern VA	Product Development
Info. Security Specialist (Associate Consultant) - DC/Northern VA	Professional Services
Product Manager - DC/Northern VA	Product Development
Security Administrator	Security Services
Senior/Principal Information Security Consultant - DC/Northern VA	Professional Services
Software Test Engineer - DC/Northern VA	Product Development
Sr. Distributed Systems Performance Tester - DC/Northern VA	Product Development
Threat Management Services Analyst - Host	Managed Services
Threat Management Services Analyst - Network	Managed Services
User Experience (UX)/Interaction Designer - DC/Northern VA	Product Development
Web Applications Developer, Server & Client Side - DC/Northern VA	Product Development
Web Front End UI Engineer - DC/Northern VA	Product Development

Linthicum, MD

Software Assurance Evaluation Engineer	

Los Angeles, CA

Info. Security Specialist (Associate Consultant) - L	
Senior/Principal Information Security Consultant - L	



New York, NY

Info. Security Manager/Director -NY	Professional Services
Info. Security Specialist (Associate Consultant) - NY	Professional Services
Senior/Principal Information Security Consultant - NY	Professional Services
User Experience (UX)/Interaction Designer - New York, NY	Product Development

Reston, VA

Configuration Management (CM) Developer Reston, VA	Product Development
Desktop UI Engineer - Reston, VA	Product Development
Product Manager - Reston, VA	Product Development
Software Test Engineer - Reston, VA	Product Development
Sr. Distributed Systems Performance Tester - Reston, VA	Product Development
User Experience (UX)/Interaction Designer - Reston, VA	Product Development
Web Applications Developer, Server & Client Side - Reston, VA	Product Development
Web Front End UI Engineer - Reston, VA	Product Development

San Francisco, CA

Info. Security Manager/Director - SF	Professional Services
Senior/Principal Information Security Consultant - SF	Professional Services

Please complete the Speaker Feedback Surveys!

Questions?

Aaron.LeMasters@Mandiant.com

@lilhoser