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

Visualization of successful heap spray in Adobe Reader (CVE-2010-2883)
The heap data map
Looking for PII

- Overlay regular expression matches on the heap map
String (unicode/ascii), byte and regex searching

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

MANDIANT is hiring!

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

**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
- Info. Security Specialist (Associate Consultant) - NY
- Senior/Principal Information Security Consultant - NY
- User Experience (UX)/Interaction Designer - New York, NY

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

**San Francisco, CA**
- Info. Security Manager/Director - SF
- Senior/Principal Information Security Consultant - SF
Please complete the Speaker Feedback Surveys!
Questions?

Aaron.LeMasters@Mandiant.com
@lilhoser