

# Using Hashing to Improve Volatile Memory Forensic Analysis

American Academy of Forensic  
Sciences Annual Meeting

February 21, 2008

**AAron Walters**

[awalters@volatilesystems.com](mailto:awalters@volatilesystems.com)

**Blake Matheny**

Volatile Systems, LLC

Center for Education and Research in Information Assurance and Security

**Doug White**

National Institute of Standards and Technology (NIST)



## **Disclaimer**

Trade names and company products are mentioned in the text or identified. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the products are necessarily the best available for the purpose.

## **Statement of Disclosure**

This research is funded in part by the National Institute of Standards and Technology.

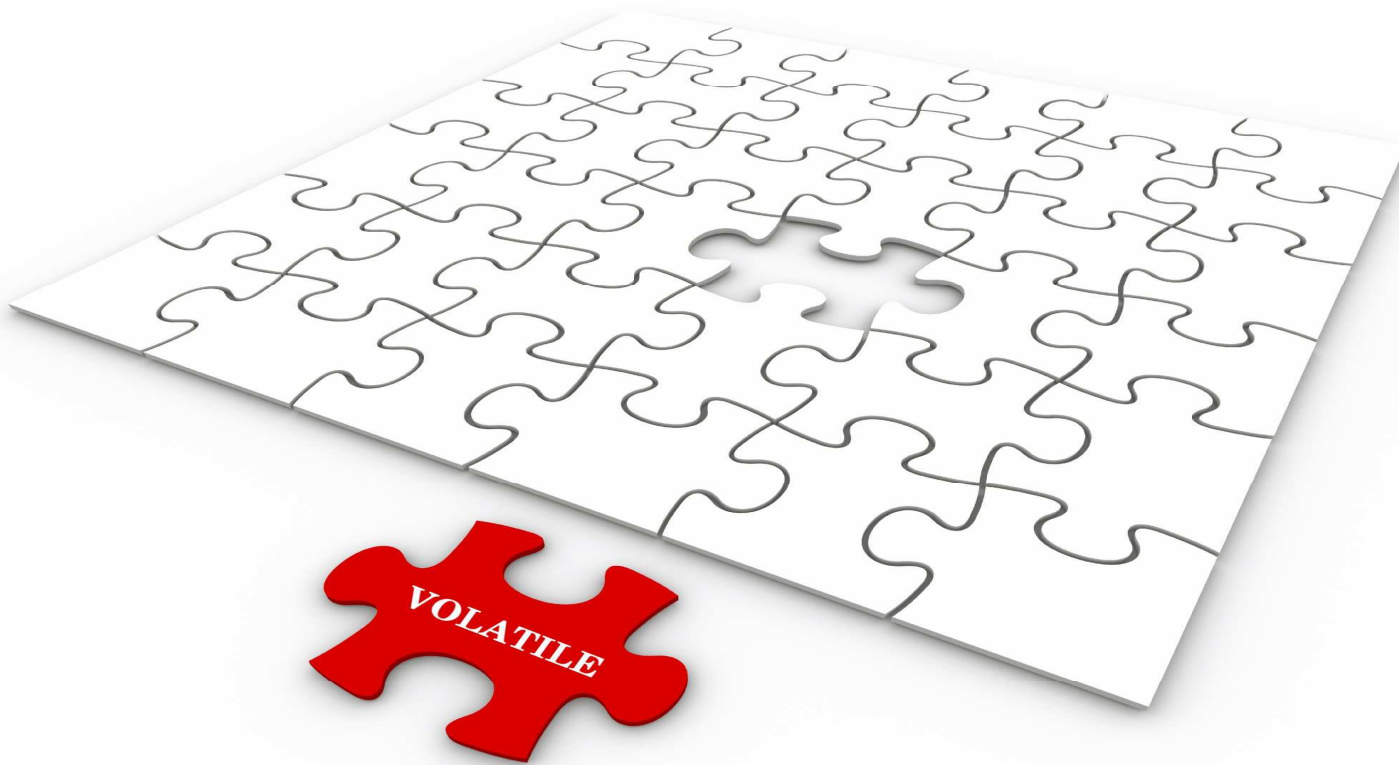
# Digital Crime Scene

---

- **Information systems**
  - Commission of crimes
  - Contain evidence related to crimes
  - Subject of crimes
- **Complex crime scene (Carrier,2003)**
  - Not just a “piece of evidence or substance”
  - Composed many types of evidence
  - Thorough analysis
  - Reconstruct crime scene

# Consistent Picture

---



# Runtime State

---

- Order of Volatility (RFC 3227)
  - Data life expectancy
- Volatile state/active objects
  - Ceases to exist when power is removed
- Valuable data (context)
- Volatile media “trusted” (pswds, keys, malware)
- Goals (Carrier, 2003):
  - Minimize obtrusiveness
  - Minimize trust
  - Understand effects

# Volatile Memory Analysis

---

- Entire contents of physical memory (RAM)
- Direct analysis of raw bit “image”
- Artifact persistence (Chow,2005)
- Advantages:
  - Analysis does not depend on OS (trust)
  - Reduce and simplify obtrusiveness (acquisition)
  - Removes the active adversary (freeze state)
  - Verifiable (3<sup>rd</sup> Party: data and tools)
  - Unconstrained analysis (raw data)
- Few trusted open resources (standard references)

# Analysis Types

**Application  
Analysis**

**Application Address Space**

**Virtual Memory  
Analysis**

**User Address Space**

**Kernel Address Space**

**Physical Memory  
Analysis**

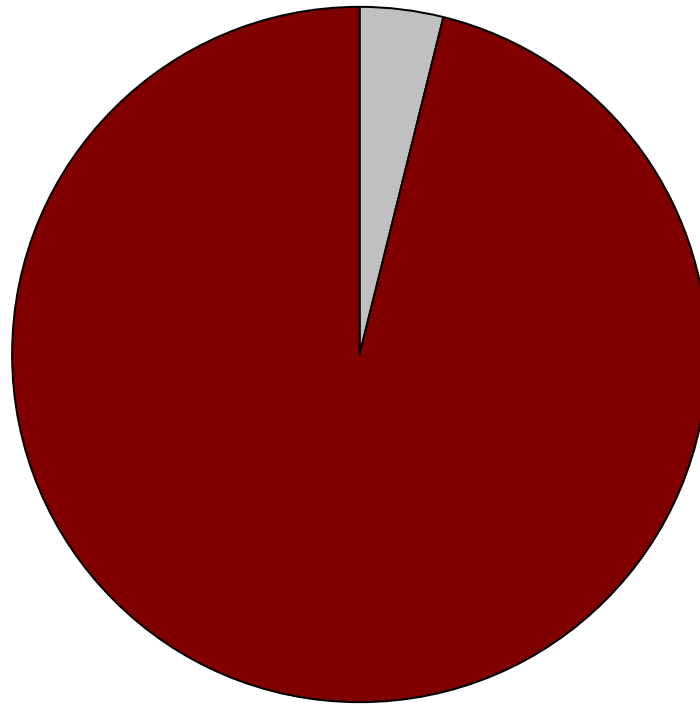
**Physical Address Space**

**Swap**

# Current State of VMA

---

- Current analysis methods based on **ad-hoc inference** analyze less than 4% of volatile storage!



# Relying on what we know

---

- **Physical crime scene (Carrier,2003)**
  - Laws of Nature
  - Familiarity and experience with physical world
- **Digital environment**
  - Hardware and software
- **Cryptographic hashes**
  - Mathematical transformation (data → hash value)
  - Uniquely identify known data
  - Detect data modifications/corruptions

# Hash Databases

---

- **File system analysis**
  - Databases of cryptographic file hashes
  - Imported by forensic applications
  - Origin and authenticity verifiable?
- **National Software Reference Library (NIST)**
  - Archive of known and traceable software
  - Reference Data Set (RDS)
  - Over 13 Million signatures
- **Can we leverage the NSRL to support VMA?**
  - Generate a standard reference data set?
  - ~ 5 Million executables

# Portable Executable (PE)

---

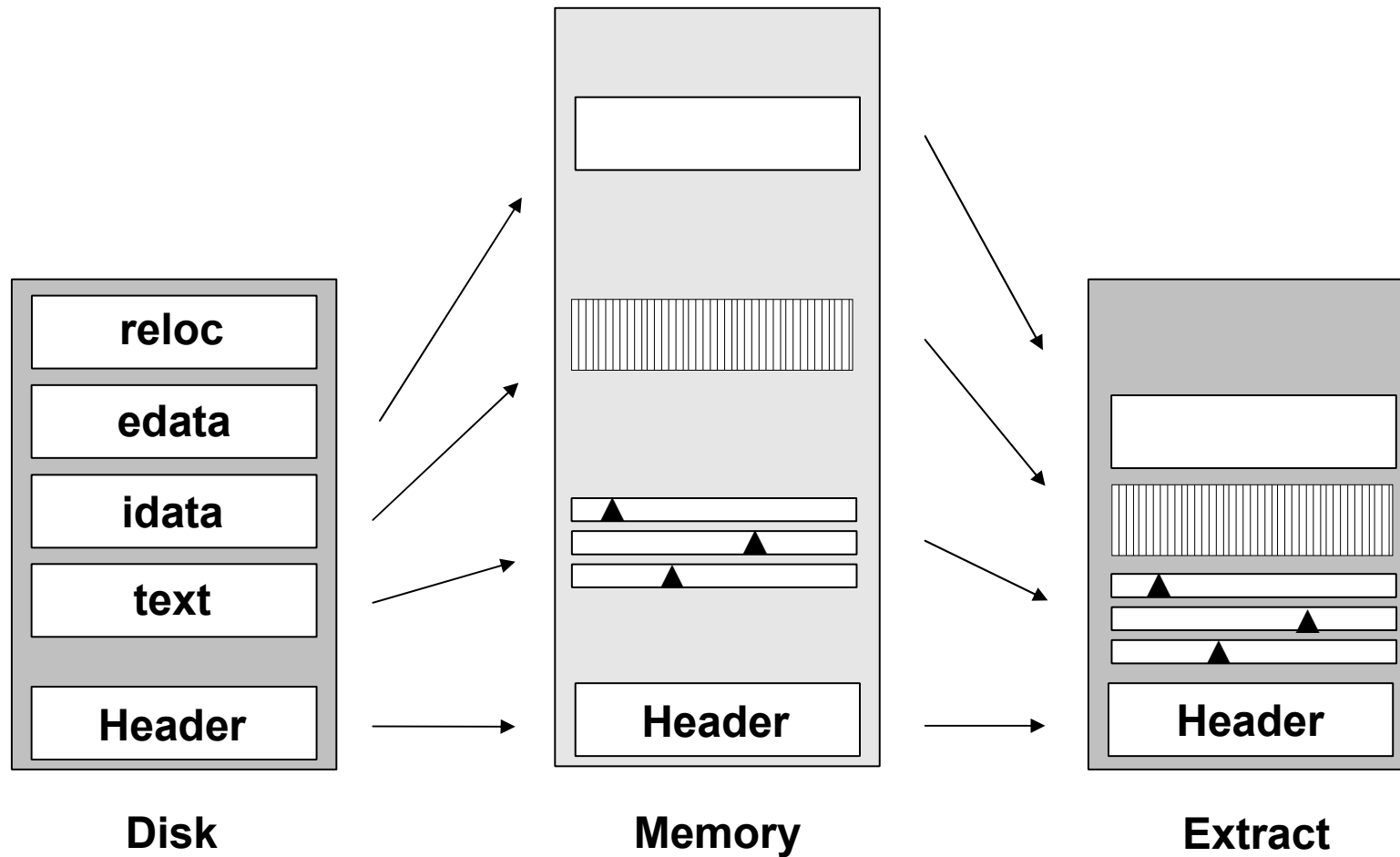
- Windows™ executable file format
  - EXE, DLL, SYS
  - Loaded into memory
- “Laws of Software”
  - Defines set of allowable code
  - Expected control flow graph (CFG)
- Find known aspects of executables in memory
  - Reduce volume of data
  - What is running (patches)?
  - Evaluate runtime integrity (Petroni, 2004)
  - Derive “chain of trust” (transitive)

# Windows™ Loader

---

- **Memory management**
  - Memory mapped (as needed)
- **Section alignment/attributes**
  - Disk vs. memory
- **Import Address Table (IAT)**
  - Library locations determined at runtime
  - Overwrites IAT entries with actual addresses
- **Base Relocations**
  - Addresses based on preferred load address
  - List of locations that need to be modified
- **Bits on disk ≠ Bits in memory**

# Disk vs. Memory



# Generating Hash Data Set

---

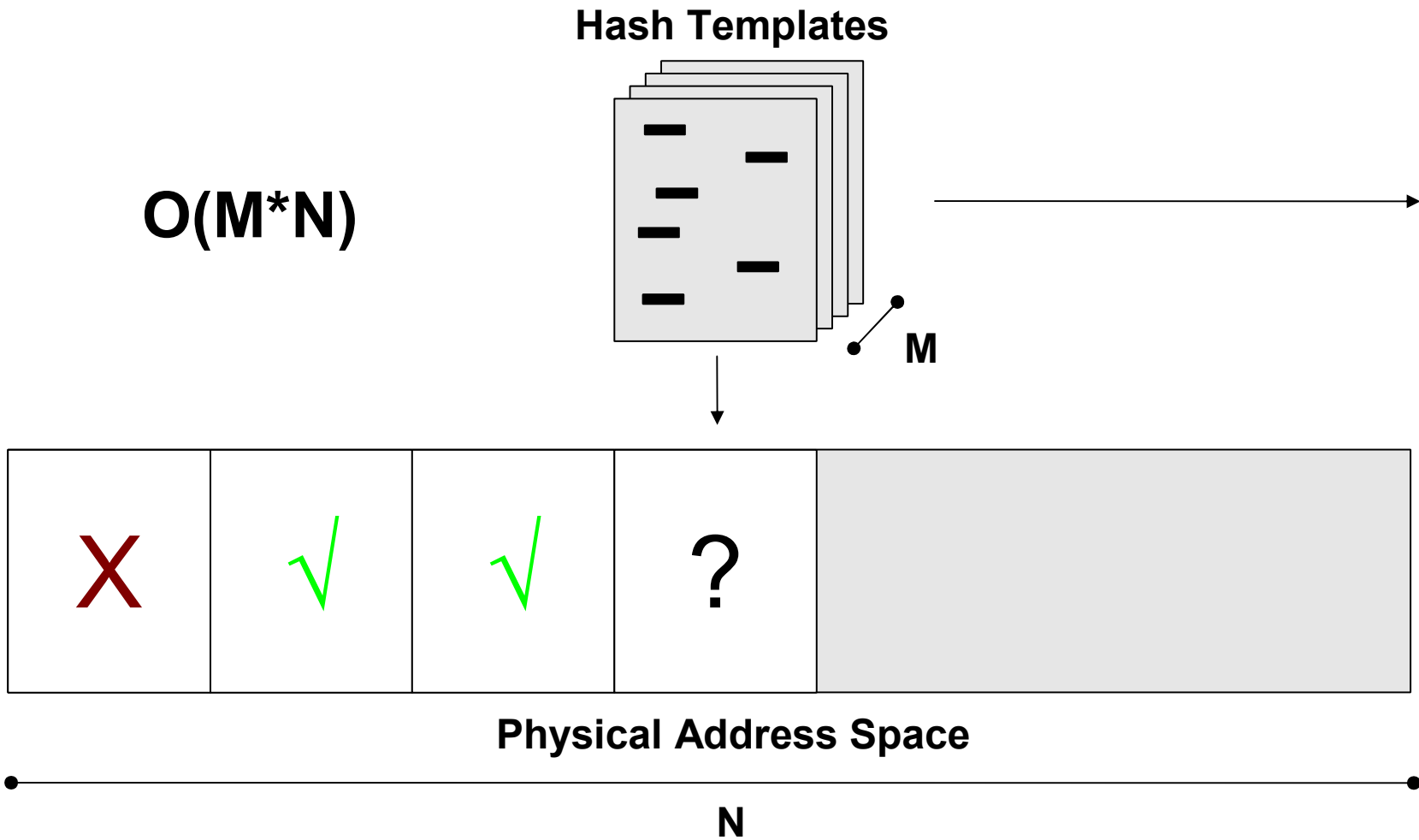
- Immutable sections of known executables
- Account for merged sections
  - Imports, etc.
- Memory align sections
  - Pages of memory
- Per page relocation templates
- Normalize relocations
- Hash on page boundaries!

# Example Hash Information

---

( 0x1000\* 0x1000\* .text\* 96f3b3ae3d51f09b08c1e4858314692843da371b\*  
023424fc26c8923162523471a28aa6ad6e4ae1b4\* [(21, 4), (55, 4), (76, 4),  
(95, 4), (147, 4), (189, 4), (240, 4), (340, 4), (462, 4), (532, 4), (546, 4), (573, 4),  
(581, 4), (626, 4), (644, 4), (678, 4), (696, 4), (767, 4), (815, 4), (824, 4), (893,  
4), (916, 4), (973, 4), (1028, 4), (1110, 4), (1135, 4), (1147, 4), (1198, 4), (1236,  
4), (1248, 4), (1289, 4), (1507, 4), (1547, 4), (1682, 4), (1692, 4), (1700, 4),  
(1706, 4), (1759, 4), (1784, 4), (1796, 4), (1801, 4), (1895, 4), (1940, 4), (2029,  
4), (2081, 4), (2094, 4), (2100, 4), (2112, 4), (2125, 4), (2131, 4), (2209, 4),  
(2254, 4), (2295, 4), (2362, 4), (2379, 4), (2408, 4), (2444, 4), (2477, 4), (2555,  
4), (2652, 4), (2693, 4), (2707, 4), (2735, 4), (2792, 4), (2825, 4), (2976, 4),  
(3027, 4), (3045, 4), (3081, 4), (3105, 4), (3194, 4), (3315, 4), (3353, 4), (3380,  
4), (3398, 4), (3404, 4), (3450, 4), (3512, 4), (3556, 4), (3563, 4), (3577, 4),  
(3598, 4), (3610, 4), (3661, 4), (3672, 4), (3764, 4), (3856, 4), (3893, 4), (3908,  
4), (3923, 4), (3940, 4), (3956, 4), (4003, 4), (4041, 4), (4068, 4), (4078, 4)] )

# Finding Known Memory



# Experimental Results

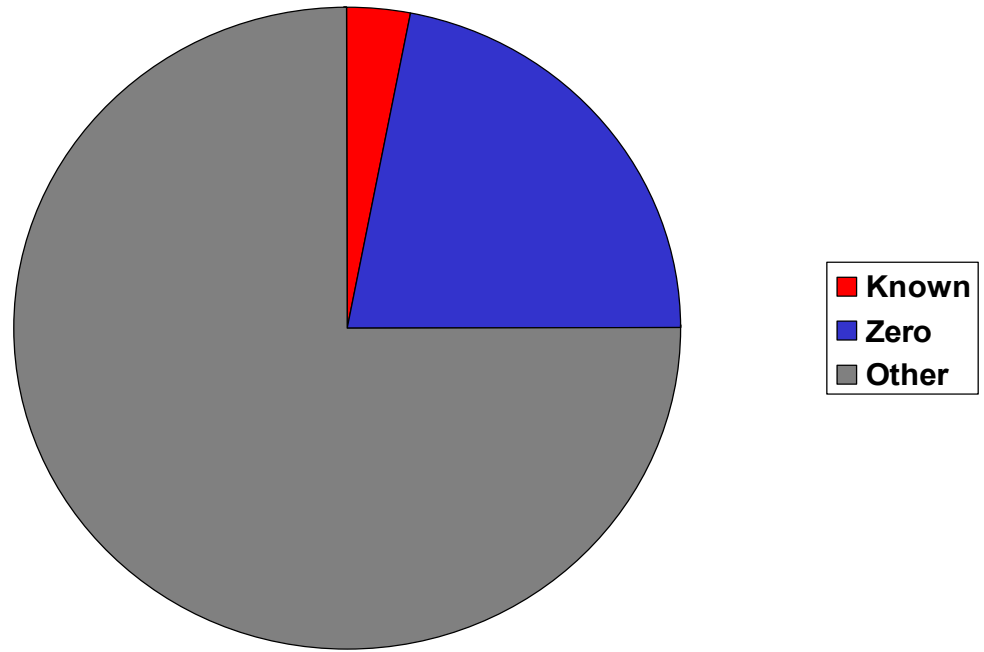
---

- **Lab: Virtual machine introspection (Garfinkel,2003)**
  - Windows™ XP Professional with Service Pack 2
  - MSDN: 2004-08-11 20:42:00 (UTC)
  - 256 MB RAM
  - 4,303 PE (99.1% RDS 2.19)
- **Real: Sampled workstation**
  - Windows™ XP Professional with Service Pack 2
  - 2 GB RAM
  - 25,450 PE (52.3% RDS 2.19)
  - Uptime: 7 days 1 hr 27 min (used: 1Y7M)

# Virtual Machine

---

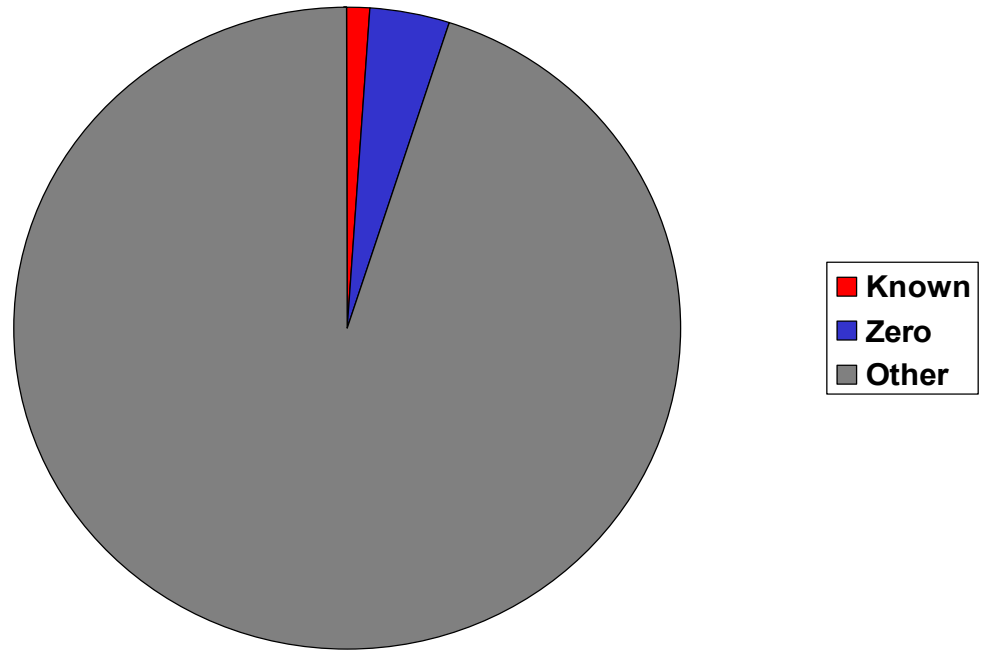
- Hash results
  - 256 MB image
  - 25%



# Real Machine

---

- Hash results
  - 2 GB image
  - >5%



# Malware

---

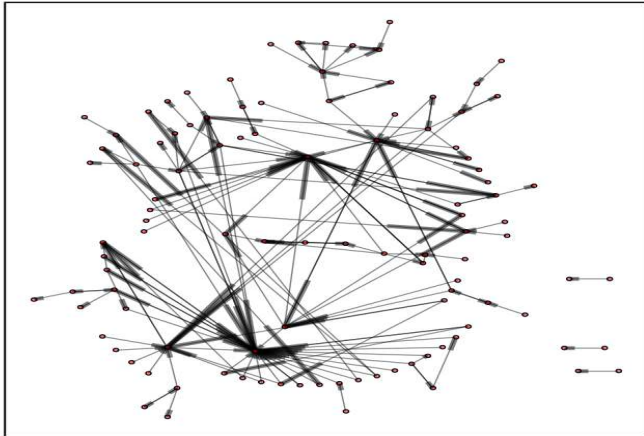
- **A/V is becoming less effective (Bailey, 2007)**
  - ~6 months → 66% detected
- **Memory resident malware/modifications**
  - Unexpected control flow changes (rootkits)
  - Anti-forensics/DLL Injection
- **Verify in-memory code sections (Petroni,SVV,GMER)**
  - History vs. file system vs. trusted hash database
  - Anchors of trust
- **Deriving trust in other data structures (transitive)**
  - Stack/Heap/etc (Walters,2006;Petroni,2007)

# eEye BootRootKit

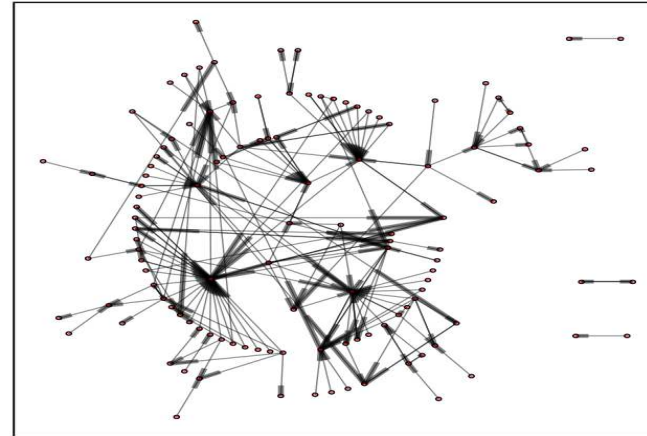
---

- Undetected by majority of rootkit detectors
  - 18 of 19 failed to detect! (Released: July 2005)
- Hooks `ethFilterDprIndicateReceivePacket`
  - Overwrites instructions with relative JMP
- Remote kernel backdoor!
- Trusted database of known hashes
  - `ndis.sys: PAGENDSE:0x500`
  - `55 8b ec 83 ec → e9 18 d5 fd ff`
  - `0xf9896b23 (05) e9 18d5fdff JMP 0xf9874040`

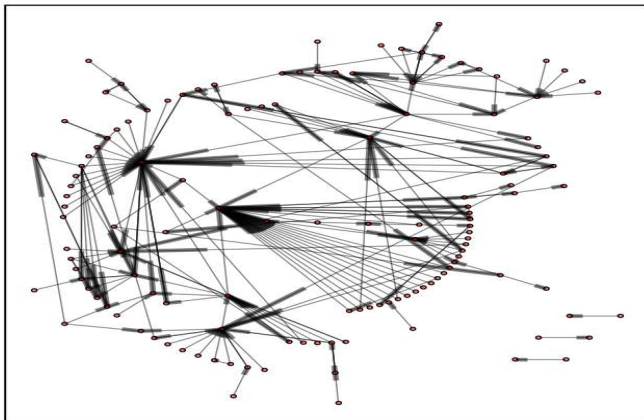
# Deriving Trust



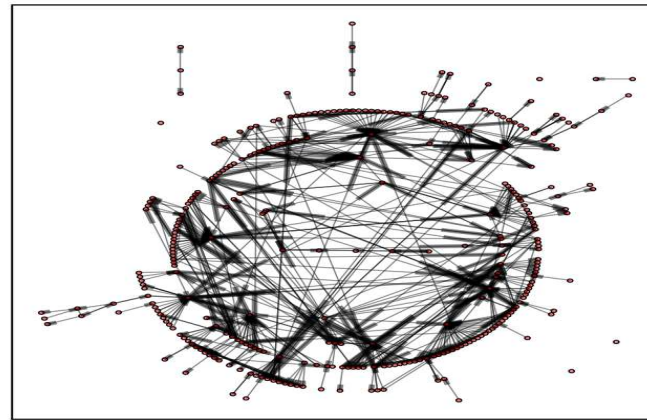
Win2000



XP



Vista



Linux

# Conclusion

---

- Volatile state is a critical component of the digital crime scene
- Build standard reference set (mimic loader)
- New public resource for VM analysts!
  - Find known memory
  - Detect malicious modifications
  - Derive trust from a potentially compromised system
- Towards complete analysis....

---

**THANK YOU!**  
**awalters@volatilesystems.com**  
**&**  
**nsrl@nist.gov**