Reverse Engineering by Crayon: Game Changing Hypervisor and Visualization Analysis

Game Changing Hypervisor Based Malware Analysis and Visualization

Danny Quist

Lorie Liebrock

New Mexico Tech Computer Science Dept.

Offensive Computing, LLC

Blackhat / Defcon USA 2009

Overview

- Reverse Engineering Process
- Hypervisors and You
- Xen and Ether
- Modifying the Process
- VERA
- Real! Live! Reversing!
- Results

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

Danny Quist

- Offensive Computing, LLC - Founder
- Ph.D. Candidate at New Mexico Tech
- Reverse Engineer
- Instructor

Lorie Liebrock

- Computer Science Department Chair, New Mexico Tech
- Associate Professor
- New Mexico Tech Scholarship for Service Principal Investigator

Overview

- **Reverse Engineering Process**
- Hypervisors and You
- Xen and Ether
- Modifying the Process
- VERA
- Real! Live! Reversing!
- Results

Process for Reverse Engineering

- Setup an isolated run-time environment
- Execution and initial analysis
- Deobfuscate compressed or packed code
- Disassembly / Code-level Analysis
- Identify and analyze relevant and interesting portions of the program

Isolated Analysis Environment

- Setup an Isolated Runtime Environment
  - Virtual machines: VMWare, Xen, KVM, …
  - Need to protect yourself from malicious code
  - Create a known-good baseline environment
  - Quickly allows backtracking if something bad happens

Execution and Initial Analysis

- **Goal**: Quickly figure out what the program is doing without looking at assembly

- Look for:
  - Changes to the file system
  - Changes to the behavior of the system
    - Network traffic
    - Overall performance
    - Ads or changed browser settings

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

Remove Software Armoring

- Program protections to prevent reverse engineering
- Done via packers – Small encoder/decoder
- Self-modifying code
- Lots of research about this
  - OllyBonE, Saffron, Polyunpack, Renovo, Ether, Azure
  - My research uses Ether

Packing and Encryption

- Self-modifying code
  - Small decoder stub
  - Decompress the main executable
  - Restore imports
- Play “tricks” with the executable
  - OS Loader is inherently lazy (efficient)
  - Hide the imports
  - Obscure relocations
  - Use bogus values for various unimportant fields

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

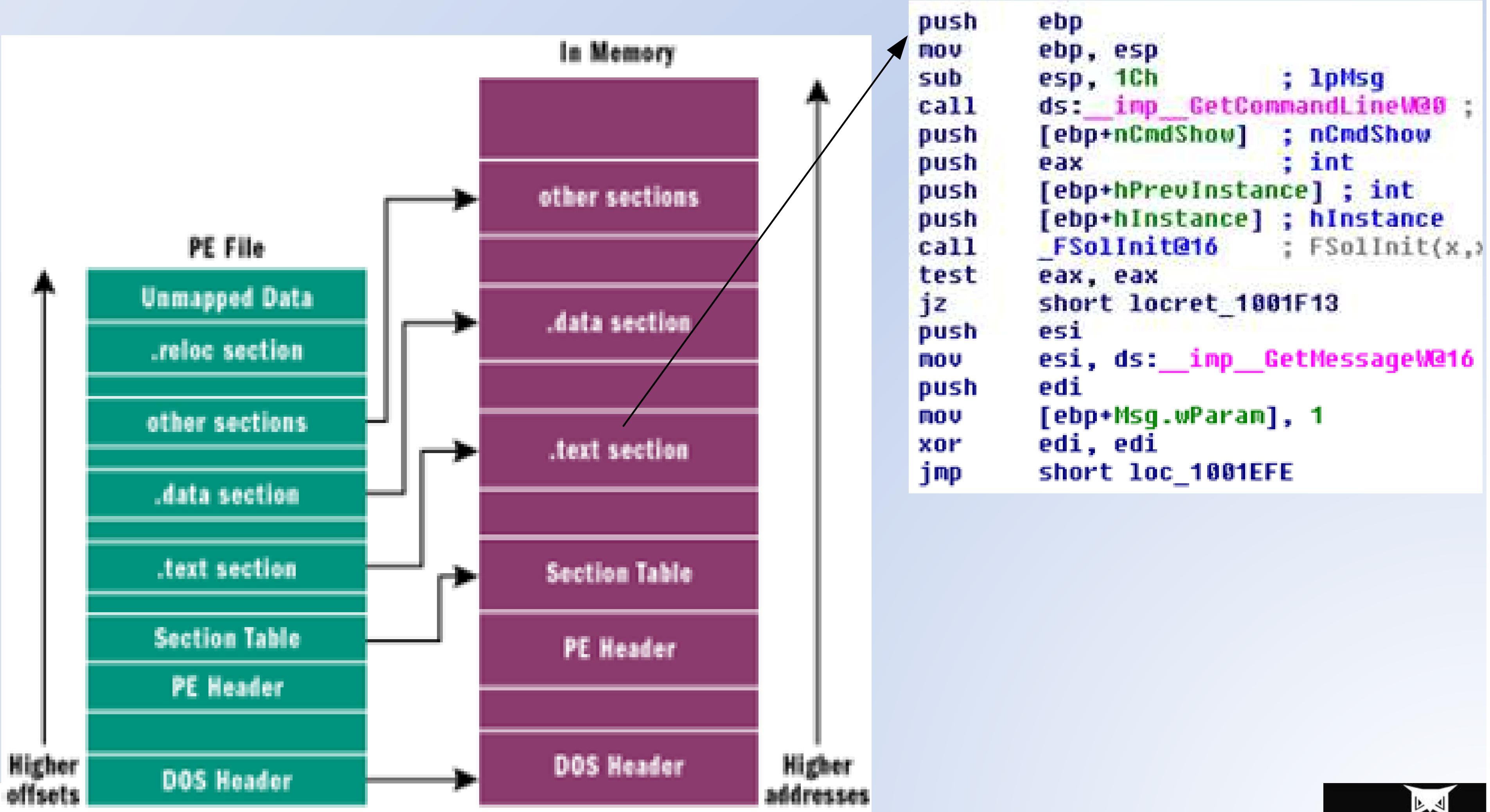
Software Armoring

- Compressed, obfuscated, hidden code
- Virtual machine detection
- Debugger detection
- Shifting decode frames

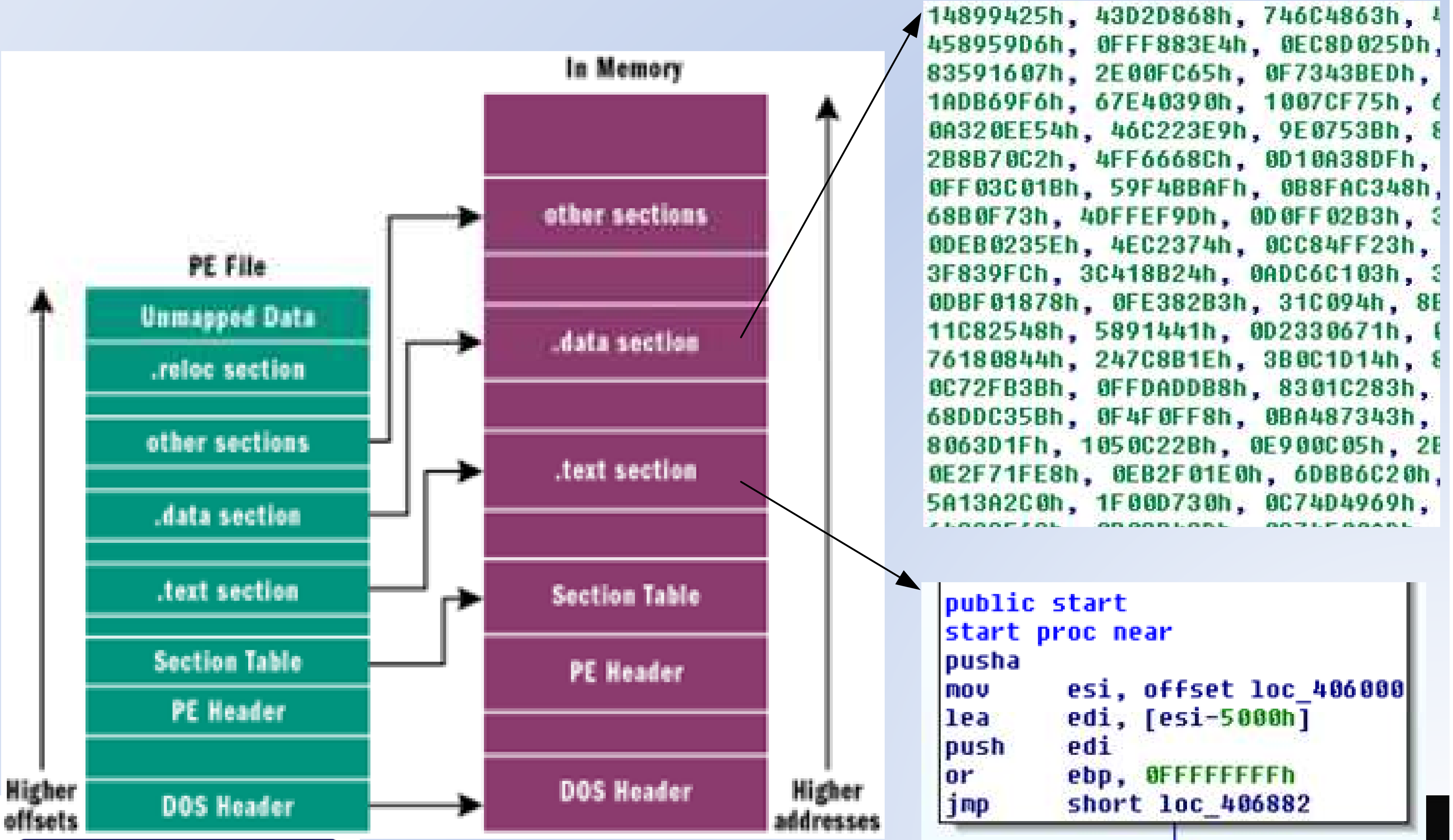
NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

Normal PE File



Packed PE File



Troublesome Protections

- Virtual Machine Detection
  - Redpill, ocvmdetect, Paul Ferrie's paper
- Debugger Detection
  - IsDebuggerPresent()
  - EFLAGS bitmask
- Timing Attacks
  - Analyze value of RDTSC before and after
  - Really effective

Thwarting Protections

Two methods for circumvention

1. Know about all the protections before hand and disable them

2. Make yourself "invisible"

Virtual Machine Monitoring

- Soft VM Based systems
  – Renovo
  – Polyunpack
  – Zynamics Bochs unpacker

- Problems
  – Detection of virtual machines is easy
  – Intel CPU never traditionally designed for virtualization
  – Do not emulate x86 bug-for-bug

OS Integrated Monitoring

- Saffron, OllyBonE
  - Page-fault handler based debugger
  - Abuses the supervisor bit on memory pages
  - High-level executions per page
- Problems
  - Destabilizes the system
  - Need dedicated hardware
  - Fine-grain monitoring not possible

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

Fully Hardware Virtualizations

- Ether: A. Dinaburg, P. Royal
  - Xen based hypervisor system
  - Base functions for monitoring
    - System calls
    - Instruction traces
    - Memory Writes
  - All interactions done by memory page mapping
- Problems
  - Old version of Xen hypervisor
  - Requires dedicated hardware

Disassembly and Code Analysis

- Most nebulous portion of the process
- Largely depends on intuition
- Looking at assembly is tedious
- Suffers from “not seeing the forest from the trees” syndrome
- Analyst fatigue – Level of attention required yields few results

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

Find Interesting and Relevant Portions of the Executable

- Like disassembly, this relies on a lot of intuition and experience
- Typical starting points:
  - Look for interesting strings
  - Look for API calls
  - Examine the interaction with the OS
- This portion is fundamentally imprecise, tedious, and often frustrating for beginners and experts

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

Overview

- Reverse Engineering Process
- **Hypervisors and You**
- Xen and Ether
- Modifying the Process
- VERA
- Real! Live! Reversing!
- Results

Hypervisors

- Lots of hype over the past few years
- New hypervisor rootkits lead defensive tools
- Covert methods for analyzing runtime behavior are extremely useful
- Detection of hardware virtualization not widely implemented

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

Useful Hypervisor Technology

- VMWare ESX Server
  - Commercial grade solution for VMs
  - Avoids VM detection issues (mostly)
- Linux Kernel Virtual Machines (KVM)
  - Separates analysis OS from target OS (slightly safer?)
  - Uses well-tested Linux algorithms for analysis
- Xen
  - Excellent set of tools for introspection
  - Uses standard QEMU image formats
  - API Controlled via Python – Integration into tools is easier

Contributions

- Modifications to Ether
  - Improve malware unpacking
  - Enable advanced tracing mechanisms
  - Automate much of the tedious portions
- Visualizing Execution for Reversing and Analysis (VERA)
  - Speed up disassembly and finding interesting portions of an executable
  - Faster identification of the Original Entry Point

Overview

- Reverse Engineering Process
- Hypervisors and You
- **Xen and Ether**
- Modifying the Process
- VERA
- Real! Live! Reversing!
- Results

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

What is Ether?

- Patches to the Xen Hypervisor
- Instruments a windows system
- Base modules available
  - Instruction tracing
  - API Tracing
  - Unpacking
- "Ether: Malware Analysis via Hardware Virtualization Extensions"
  Dinaburg, Royal, Sharif, Lee

  ACM CCS 2008

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

Ether Event Tracing

- Detects events on an instrumented system
  - System call execution
  - Instruction execution
  - Memory writes
  - Context switches

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

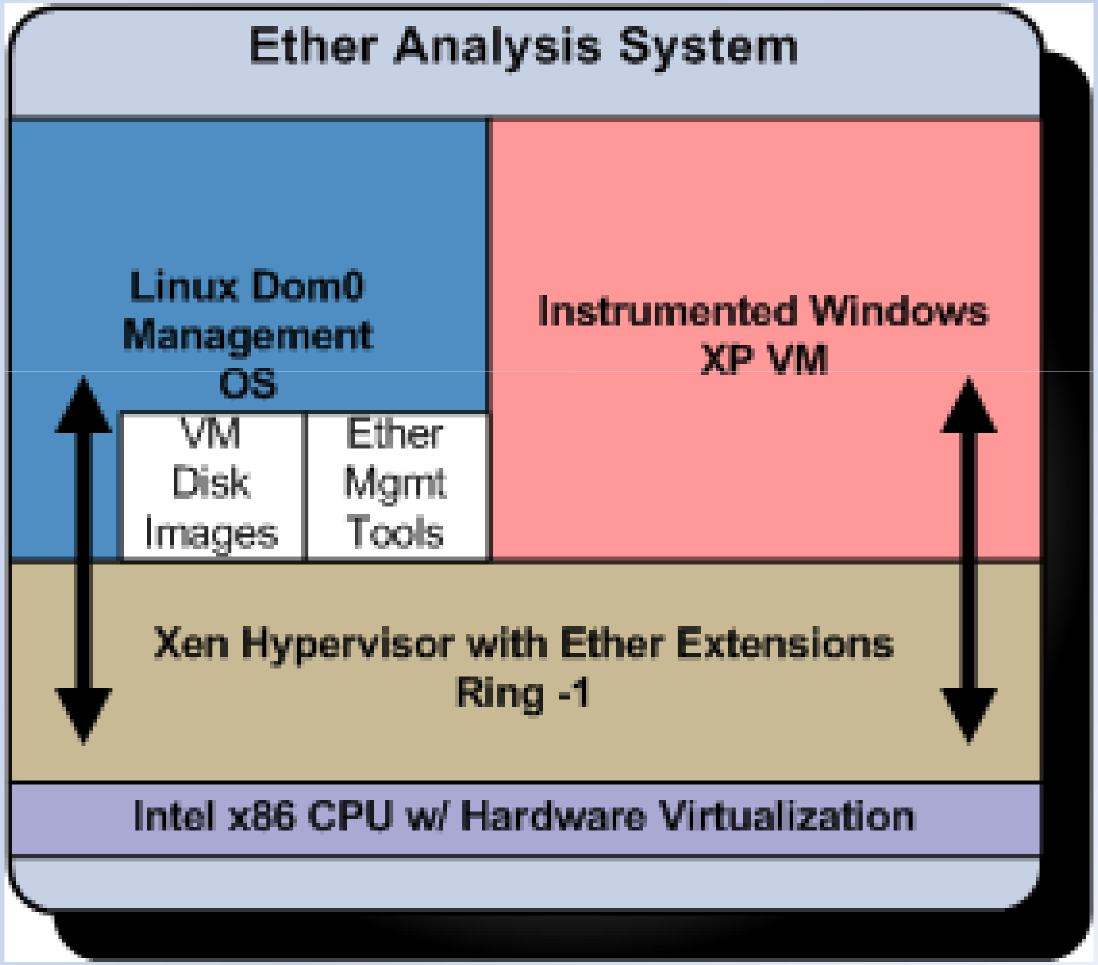
Instruction Tracing

- EFLAGS register modified for single-step (trap flag)
- PUSHF and POPF instructions are intercepted
- Modifications to this single-stepping effectively hidden (except

Memory and System Calls

- Memory Writes
  - Tracked by manipulating the shadow page table
  - Gives access to the written and read memory addresses
- System Calls
  - Modifies the SYSENTER_EIP register to point to non-paged address space
  - Logged, returned to ether
  - Overrides 0x2e interrupt to catch older syscalls

Ether System Architecture



Extensions to Ether

- Removed unpacking code from hypervisor into user-space
- Better user mode analysis
- PE Repair system – Allows for disassembly of executables
- Added enhanced monitoring system for executables

User mode Unpacking

- Watch for and monitor all memory writes
- Allow program to execute
- When execution occurs in written memory, dump memory
- Each dump is a candidate for the OEP
- Not perfect, but very close
- Scaffolding for future modifications

PE Repair

- Dumped PE files had problems
  - Sections were not file aligned
  - Address of Entry Point invalid
  - Would not load in IDA correctly
- Ported OllyDump code to Ether user mode
  - Fix section offsets to match data on disk
  - Repair resources as much as possible
  - Set AddressOfEntryPoint to be the candidate OEP

Results

- Close to a truly covert analysis system
  - Ether is nearly invisible
  - Still subject to bluepill detections
- Fine-grain resolution of program execution
- Application memory monitoring and full analysis capabilities
- Dumps from Ether can now be loaded in IDA Pro without modification

Ether Unpacking Demo!

Open Problems

- Unpacking process produces lots of candidate dump files
- Better Original Entry Point discovery method
- Import rebuilding is still an issue
- Now that there is a nice tool for tracing programs covertly, we need to do analysis

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

Overview

- Reverse Engineering Process
- Hypervisors and You
- Xen and Ether
- **Modifying the Process**
- VERA
- Real! Live! Reversing!
- Results

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

Modifying the Process

- Knowing what to look for is often the portion that most new reversers have trouble with
- Having an idea of the execution flow of a program is extremely useful
  - IDA is focused on the function view
  - Extend to the basic block view
- Software armoring removal made easy

Visualization of Trace Data

- Goals:
  - Quickly visually subvert software armoring
  - Identify modules of the program
    - Initialization
    - Main loops
    - End of unpacking code
  - Figure out where the self-modifying code ends (OEP detection)
  - Discover dynamic runtime program behavior
  - Integrate with existing tools

Visualizing the OEP Problem

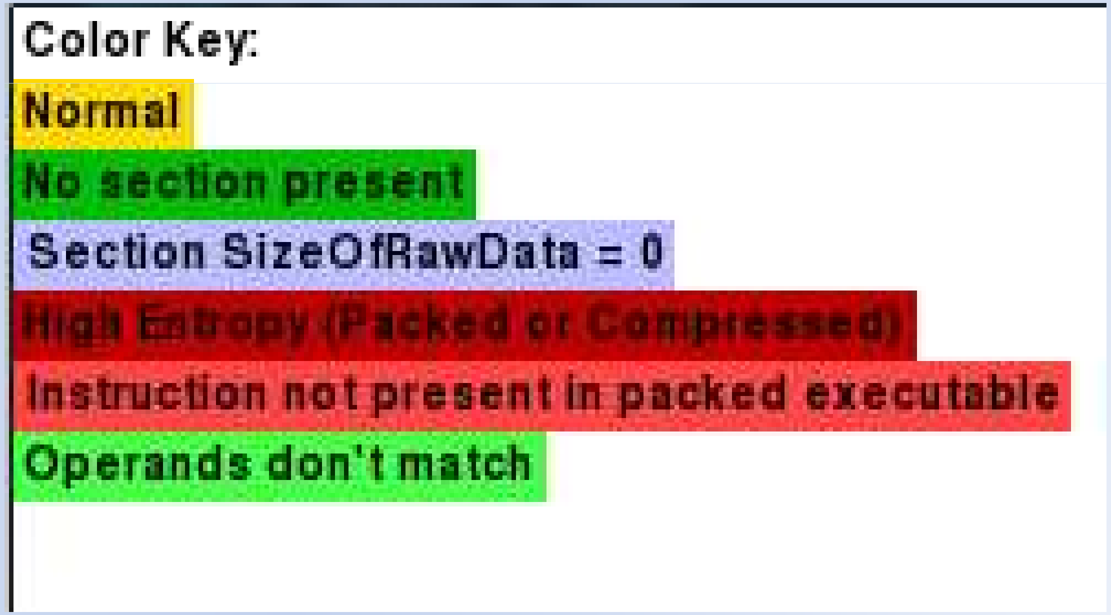
- Each block (vertex) represents a basic block executed in the user mode code
- Each line represents a transition
- The thicker the line, the more it was executed
- Colors represent areas of memory execution

VERA

- Visualization of Executables for Reversing and Analysis
- Windows MFC Application
- Integrates with IDA Pro
- Fast, small memory footprint

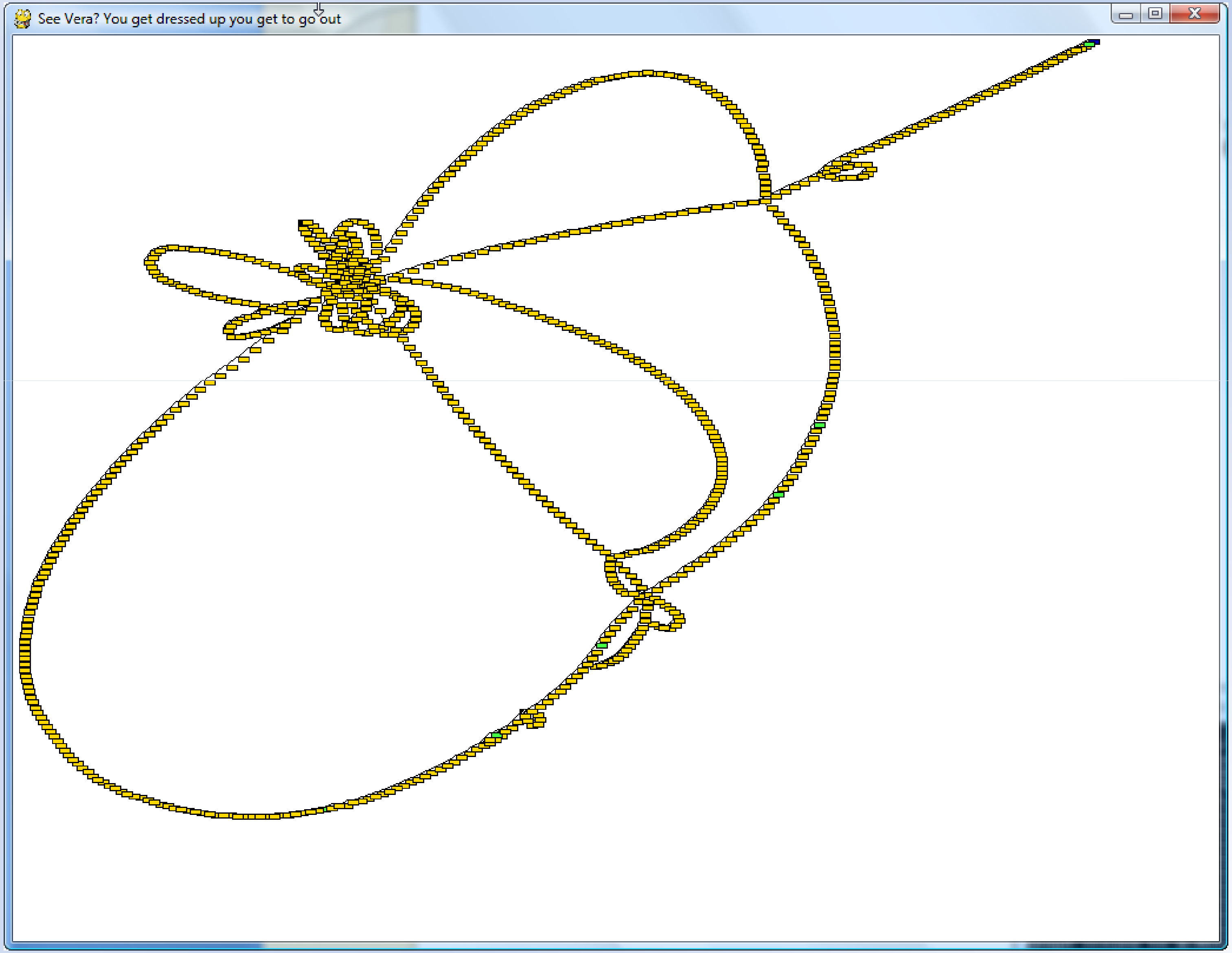
Visualizing Packers

- Memory regions marked for PE heuristics

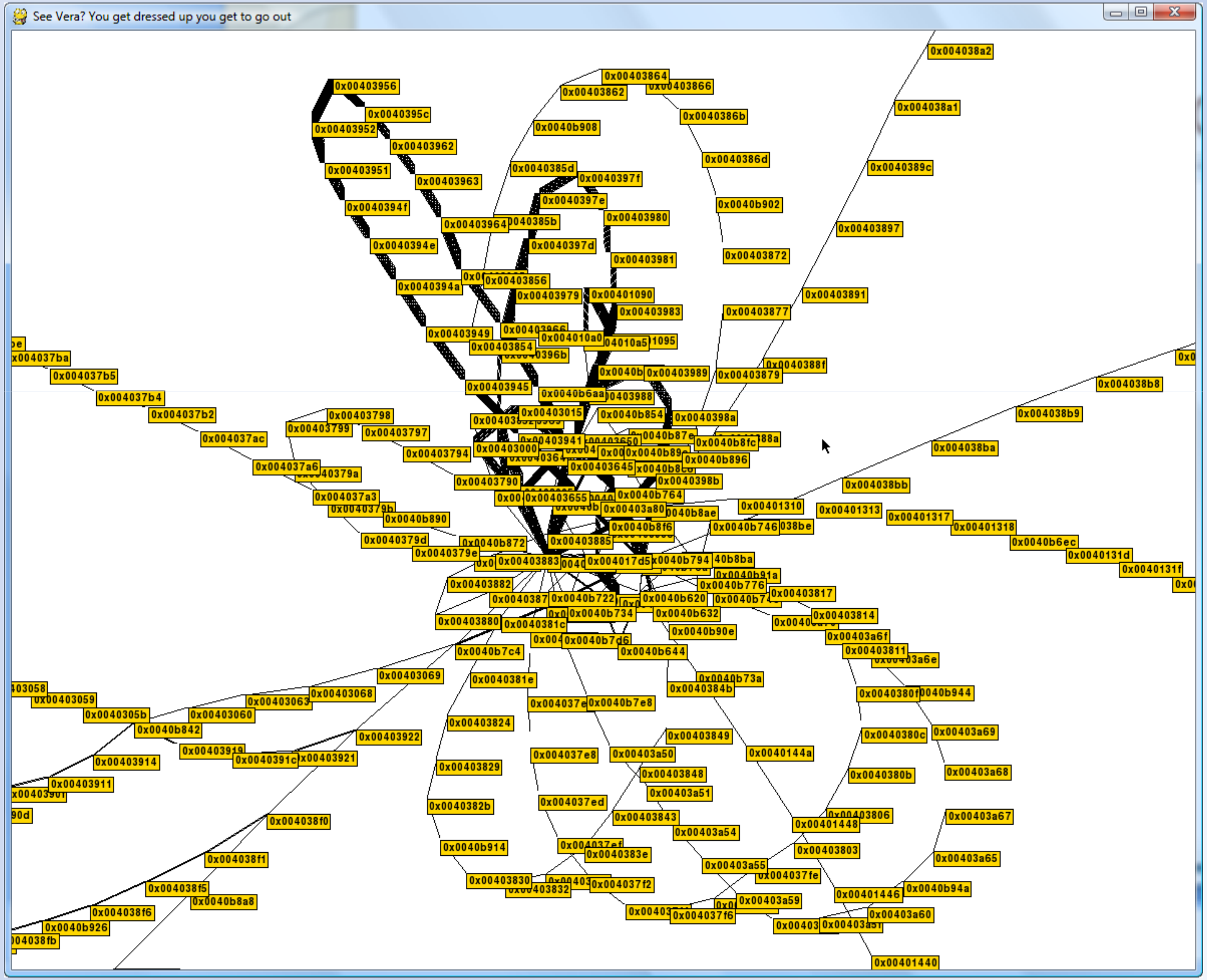


Demo!

Netbull Virus (Not Packed)

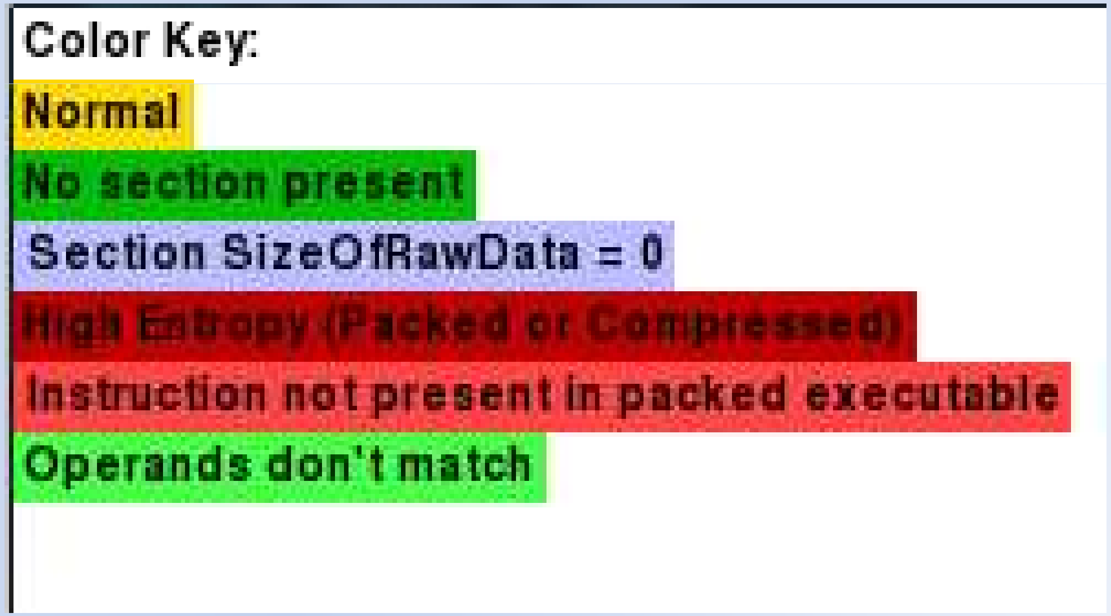


Netbull Zoomed View



Visualizing Packers

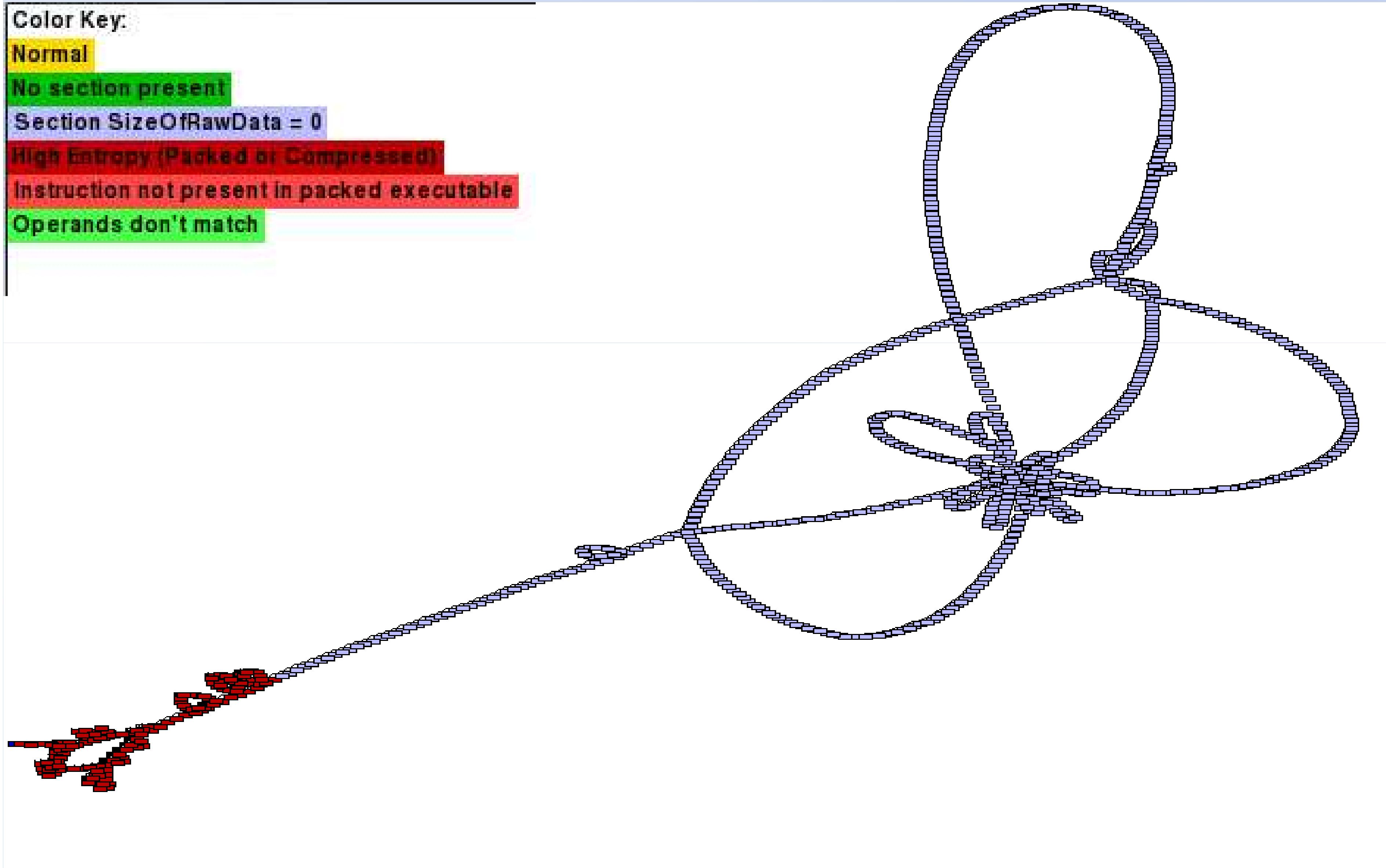
- Memory regions marked for PE heuristics



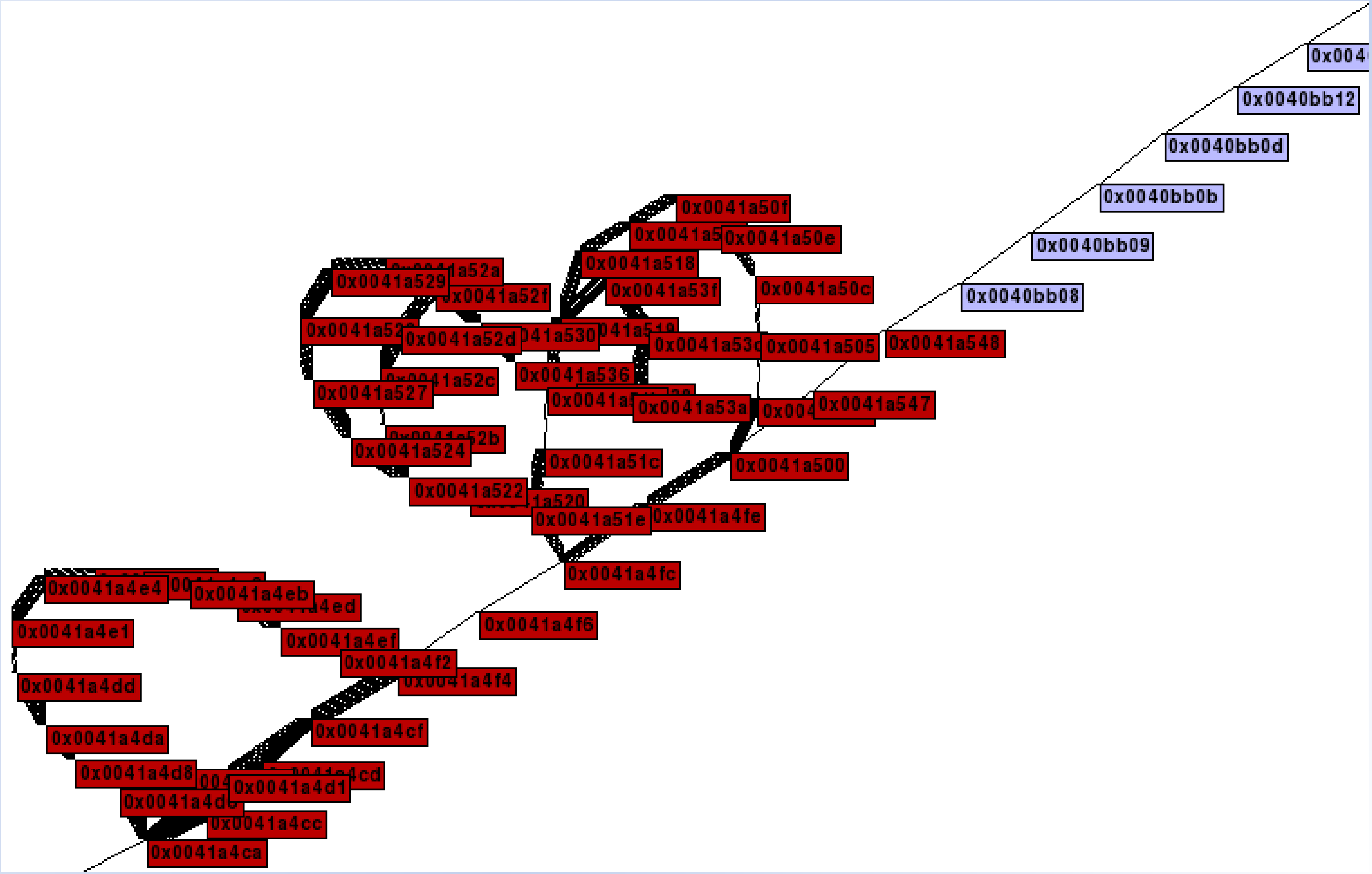
NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

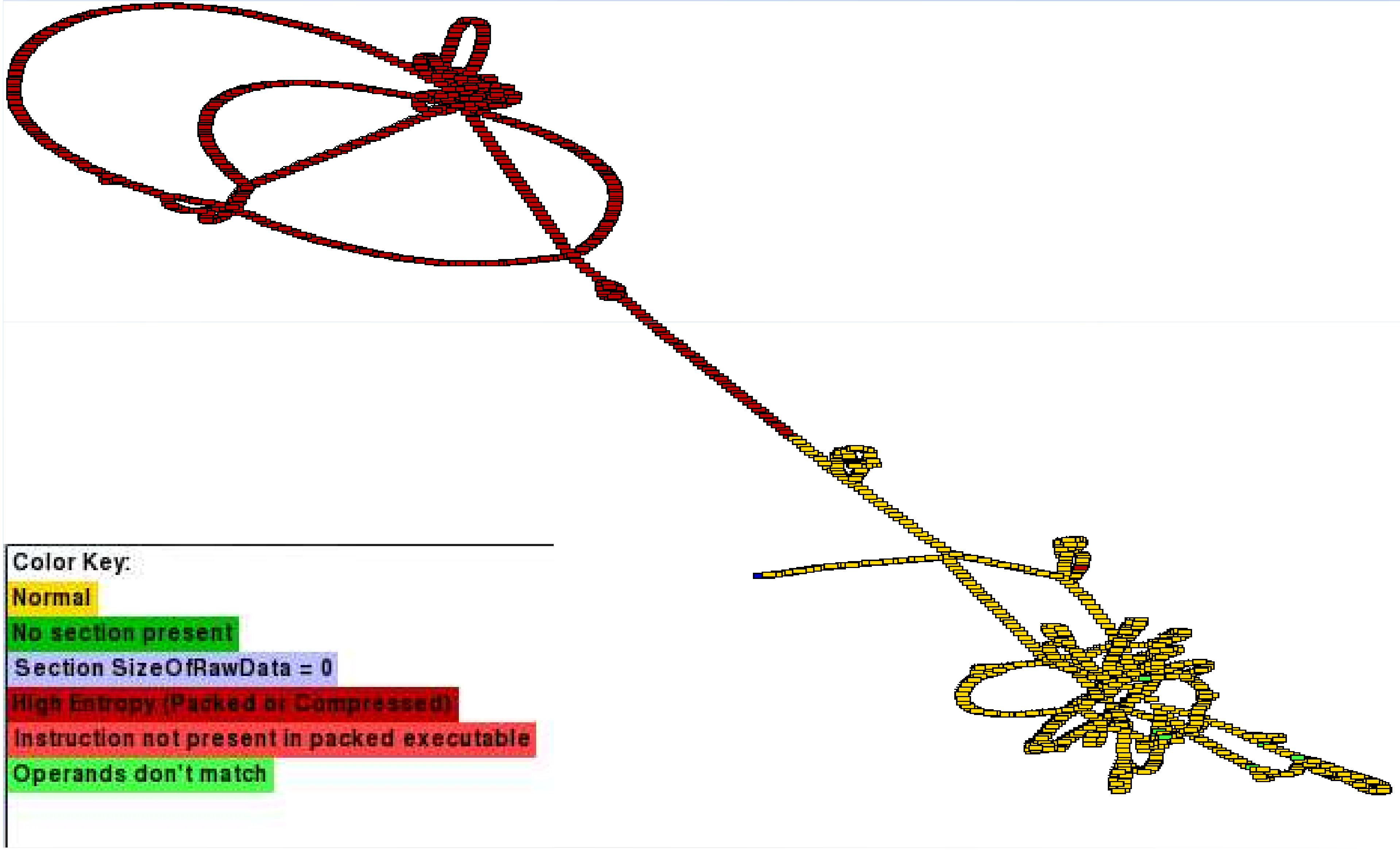
UPX


Color Key:
Normal
No section present
Section SizeOfRawData = 0
High Entropy (Packed or Compressed)
Instruction not present in packed executable
Operands don't match

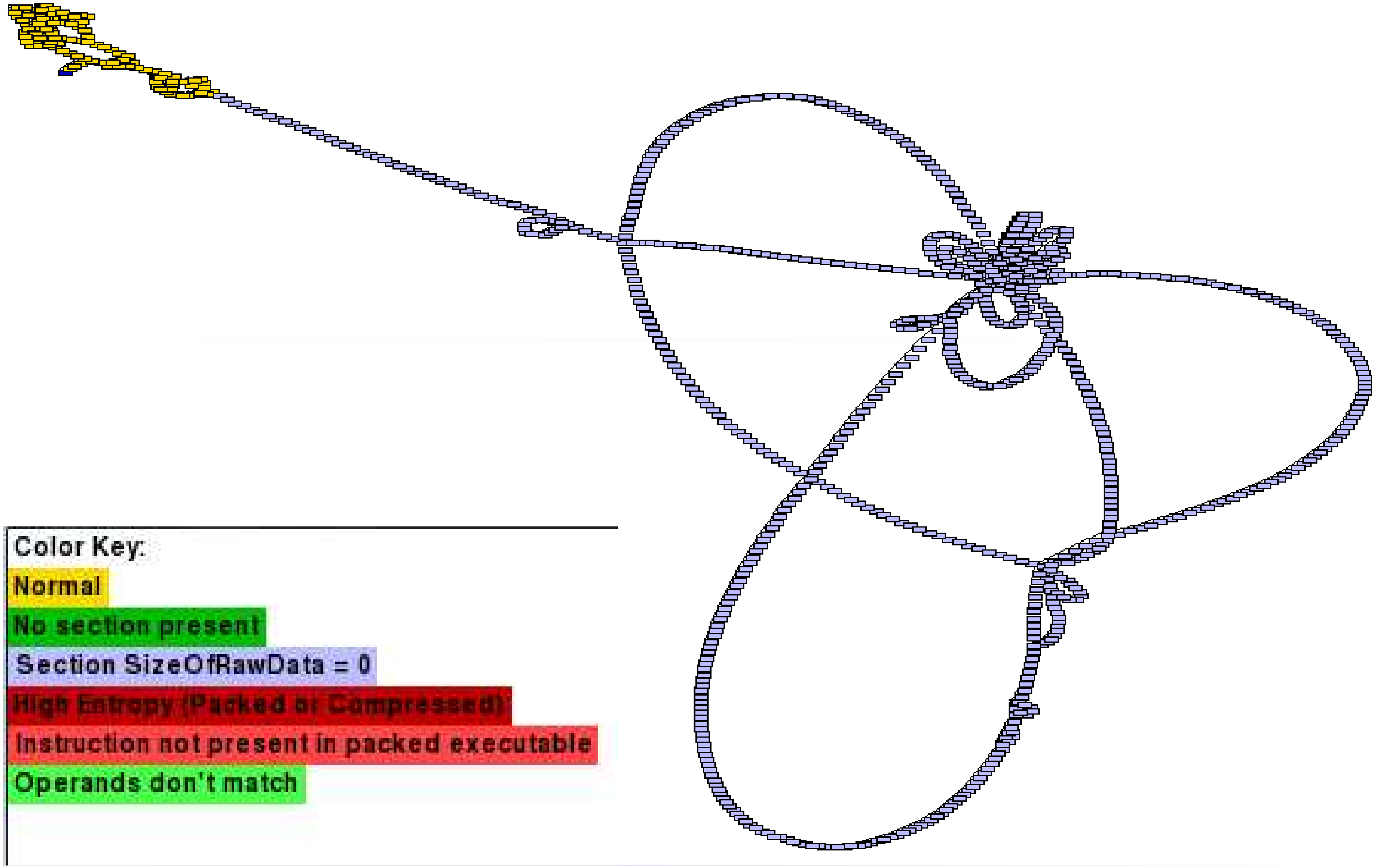
UPX - OEP


0x0040bb12
0x0040bb0d
0x0040bb0b
0x0040bb09
0x0040bb08
0x0041a50f
0x0041a50e
0x0041a518
0x0041a53f
0x0041a50c
0x0041a529
0x0041a52f
0x0041a52d
0x0041a536
0x0041a505
0x0041a548
0x0041a527
0x0041a53a
0x0041a547
0x0041a524
0x0041a51c
0x0041a500
0x0041a522
0x0041a51e
0x0041a4fe
0x0041a4fc
0x0041a4f6
0x0041a4e4
0x0041a4eb
0x0041a4e1
0x0041a4ef
0x0041a4f2
0x0041a4dd
0x0041a4cf
0x0041a4da
0x0041a4d8
0x0041a4d1
0x0041a4cc
0x0041a4ca

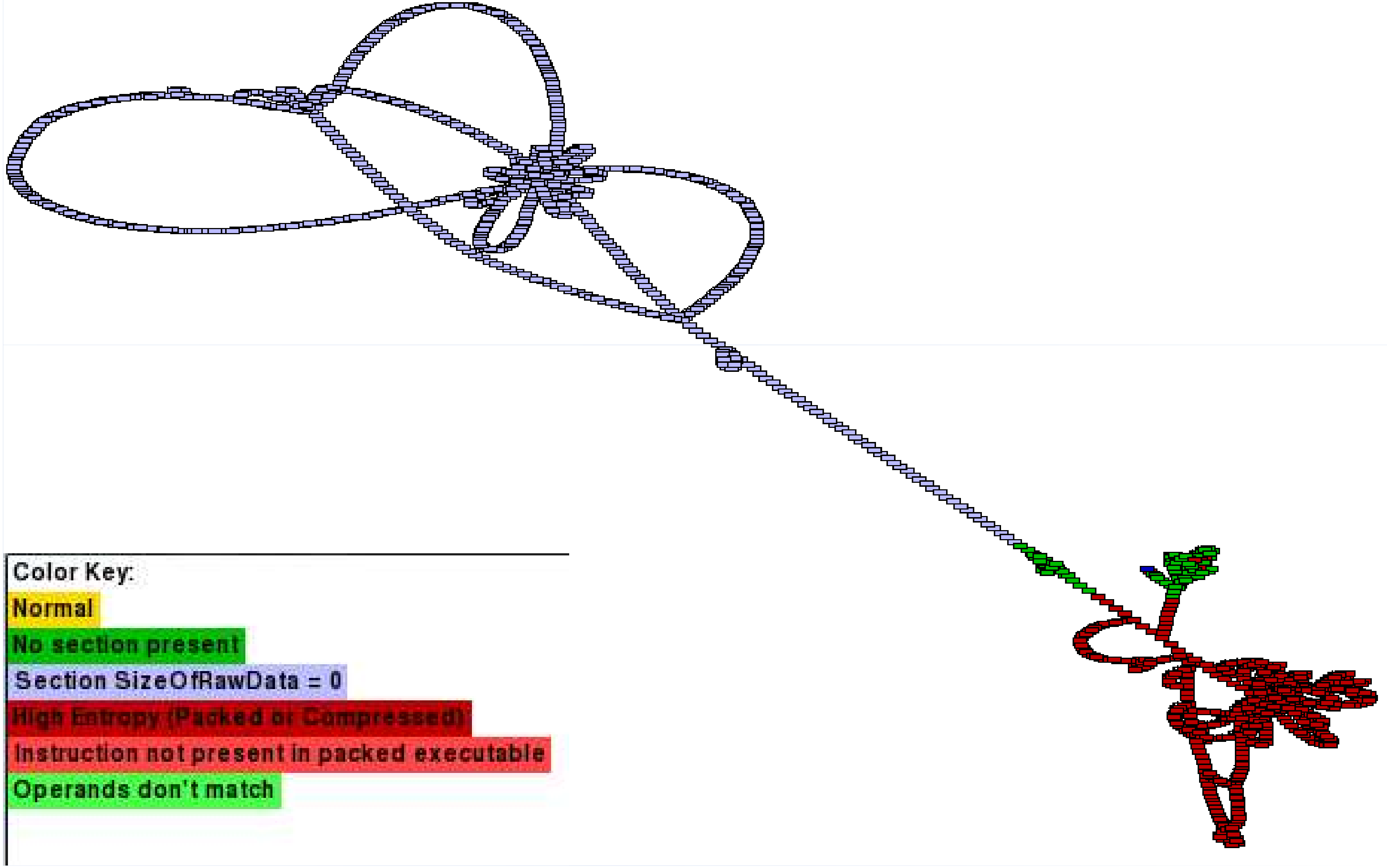
ASPack


Color Key:
Normal
No section present
Section SizeOfRawData = 0
High Entropy (Packed or Compressed)
Instruction not present in packed executable
Operands don't match

FSG

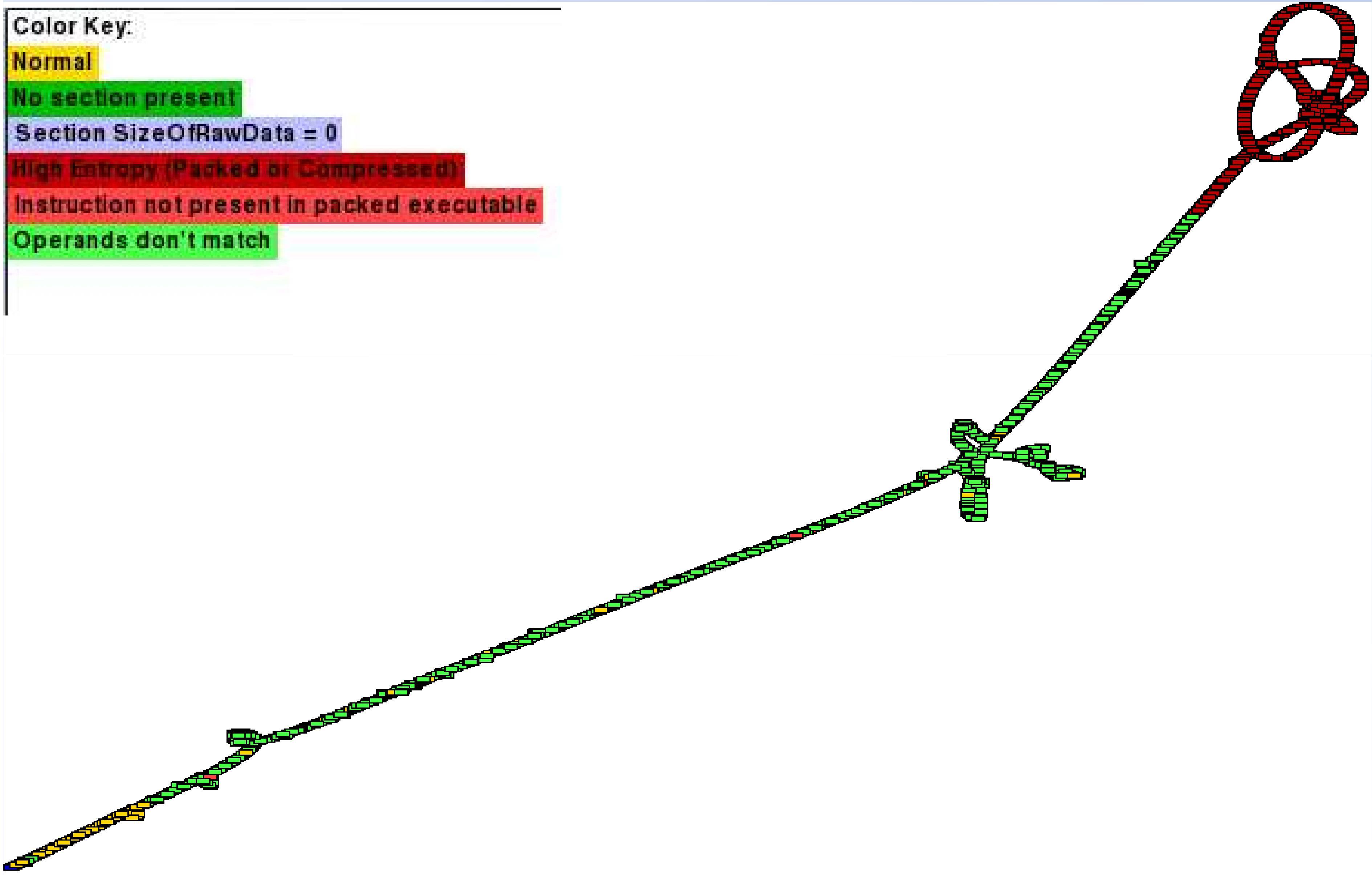

Color Key:
Normal
No section present
Section SizeOfRawData = 0
High Entropy (Packed or Compressed)
Instruction not present in packed executable
Operands don't match

MEW



Color Key:
Normal
No section present
Section SizeOfRawData = 0
High Entropy (Packed or Compressed)
Instruction not present in packed executable
Operands don't match

TeLock


Color Key:
Normal
No section present
Section SizeOfRawData = 0
High Entropy (Packed or Compressed)
Instruction not present in packed executable
Operands don't match

User Study

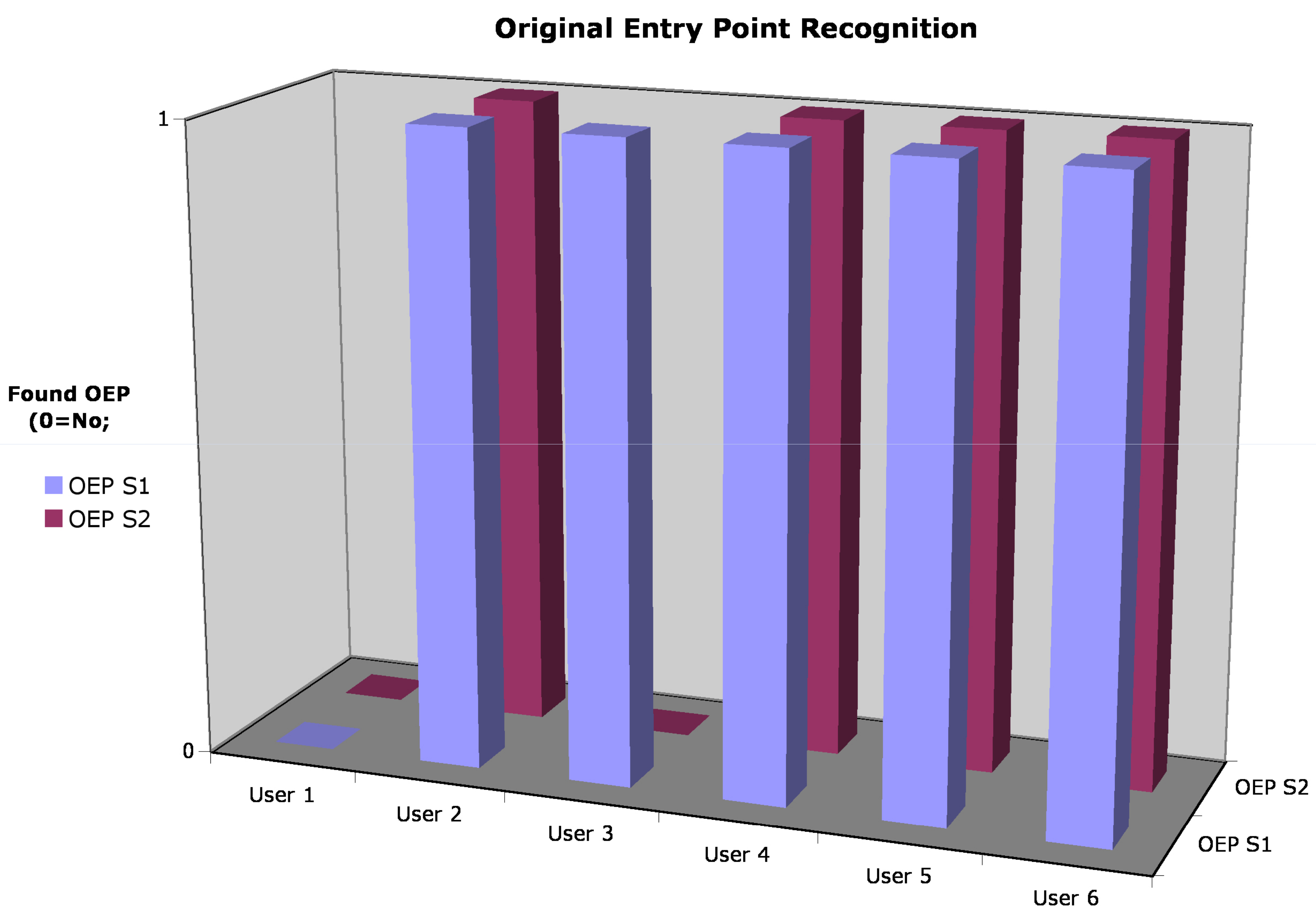
- Students had just completed week long reverse engineering course
- Analyzed two packed samples of the Netbull Virus with UPX and MEW
- Asked to perform a series of tasks based on the typical reverse engineering process
- Asked about efficacy of visualization tool

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

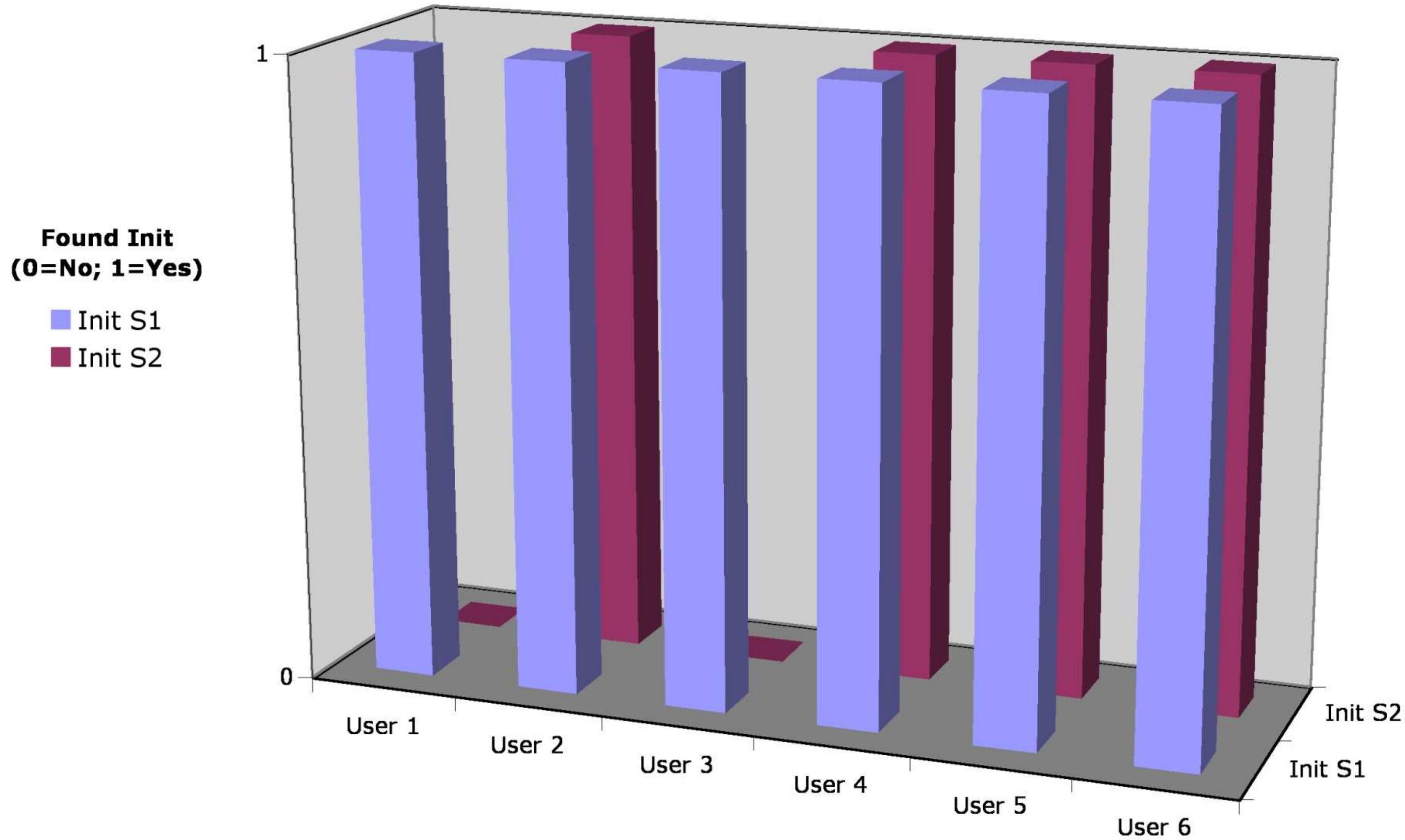
offensive computing

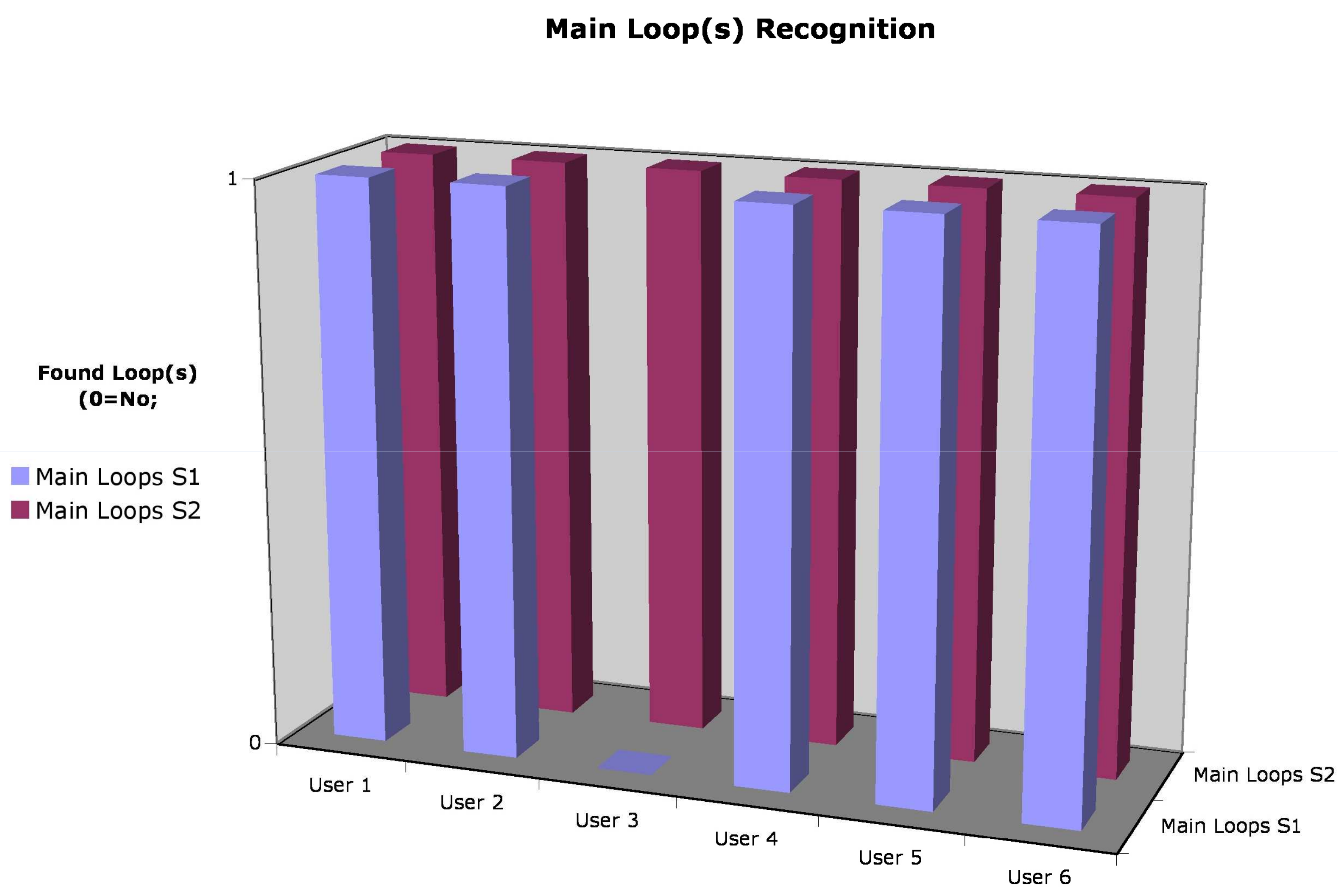
User Study: Tasks Performed

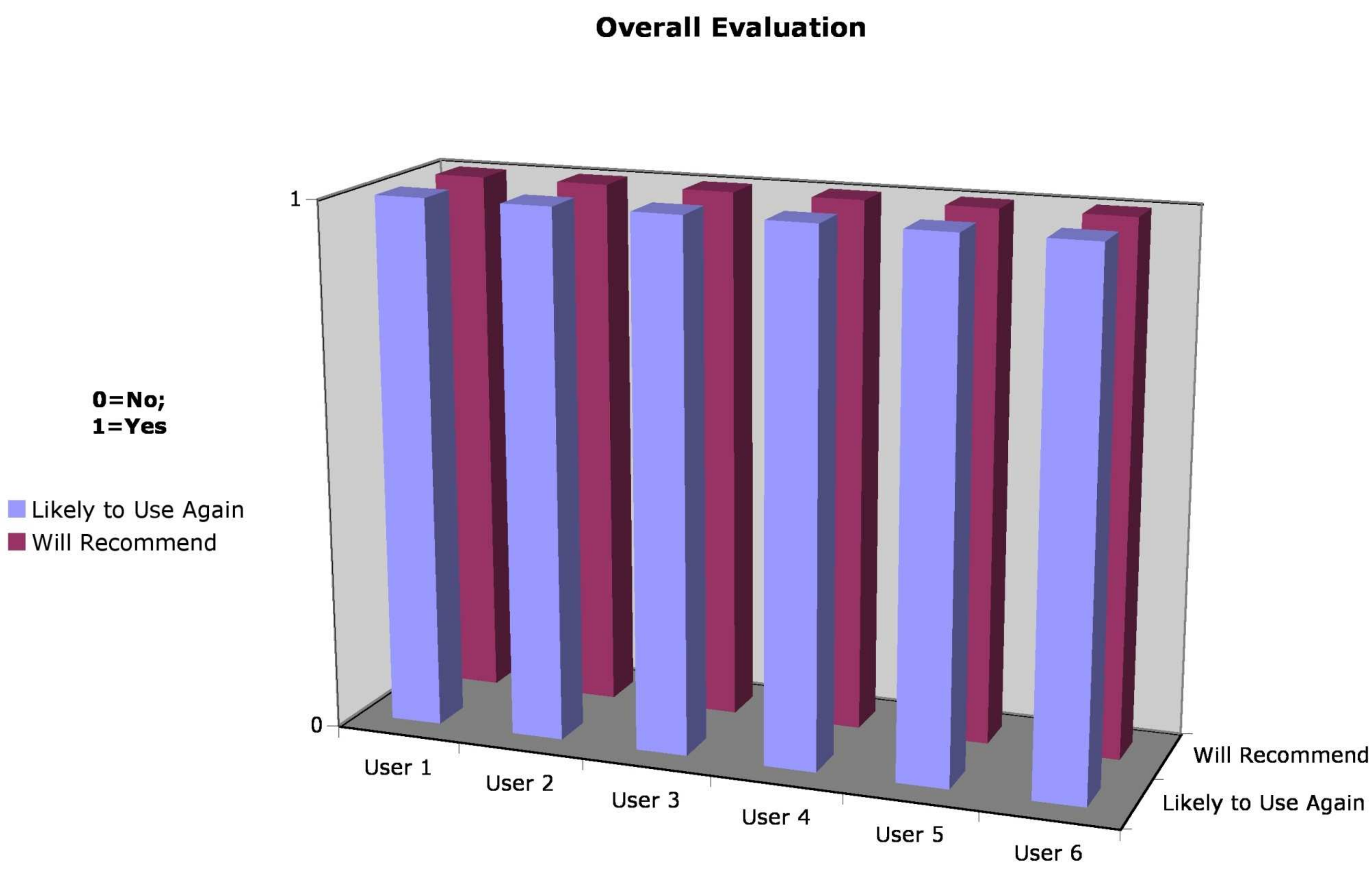
- Find the original entry point (OEP) of the packed samples
- Execute the program to look for any identifying output
- Identify portions of the executable:
  - Packer code
  - Initialization
  - Main loops


Original Entry Point Recognition
1
0
Found OEP
(0=No;
OEP S1
OEP S2
User 1
User 2
User 3
User 4
User 5
User 6
OEP S2
OEP S1

Initialization Recognition




Main Loop(s) Recognition
Found Loop(s)
(0=No;
Main Loops S1
Main Loops S2
1
0
User 1
User 2
User 3
User 4
User 5
User 6
Main Loops S2
Main Loops S1


Overall Evaluation
0=No;
1=Yes
Likely to Use Again
Will Recommend
1
0
User 1
User 2
User 3
User 4
User 5
User 6
Will Recommend
Likely to Use Again

Selected Comments

- “Wonderful way to visualize analysis and to better focus on areas of interest”
- “Fantastic tool. This has the potential to significantly reduce analysis time.”
- “It rocks. Release ASAP.”

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

Recommendations for improvement

- Need better way to identify beginning and end of loops
- Many loops overlap and become convoluted
- Be able to enter memory address and see basic blocks that match

Future Work

- General GUI / bug fixes
- Memory access visualization
- System call integration
- Function boundaries
- Interactivity with unpacking process
- Modify hypervisor to work with WinDBG, OllyDbg, IDA Debugger

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

Conclusions

- Visualizations make it easy to identify the OEP
- No statistical analysis of data needed
- Program phases readily identified
- Graphs are relatively simple
- Preliminary user study shows tool holds promise for speeding up reverse engineering

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH • UNIVERSITY

offensive computing

Thanks!

- Artem Dinaburg
- Paul Royal
- Cort Dougan
- Moses Schwartz
- Alan Erickson
- Alex Kent
- New Mexico Tech SFS Program

Closing thoughts

- Ether is awesome. Thanks Artem Dinaburg and Paul Royal.
- Source, tools, and latest slides can be found at: http://www.offensivecomputing.net
- If you use the tool, please give feedback
- Look for the paper at Vizsec 2009