

Temporal Reverse Engineering



Danny Quist
Colin Ames

Offensive Computing, LLC

Blackhat USA 2008

Danny Quist

- Co-founder Offensive Computing, LLC
- Ph.D. Candidate at New Mexico Tech
- Senior Instructor – Reverse Engineering
Infosec Institute
- dquist@offensivecomputing.net

Colin Ames

- Security Researcher, Offensive Computing
- Steganography Research
- Penetration Testing
- Reverse Engineering
- Malware Analysis
- amesc@offensivecomputing.net

Offensive Computing, LLC

- Malware Community
 - Free access to malware samples
 - Largest open malware site on the Internet
 - ~1million hits per month
- Business Services

Temporal Reverse Engineering

- Observe program change over time
- Analysis is very time dependant
- Tools developed to make the time analysis easier
- Lower complexity for analysis

Overview of Talk

- Current Reverse Engineering Techniques
 - Where they work
 - Where they fail
- What is Temporal Reverse Engineering?
- Techniques and Strategies
- Checkpointing as an analysis tool
- Visualization Methods
- Applications and Demos

Reverse Engineering

- Reversing is hard
- Goal: Figure out how program works in minimal amount of time
- Expensive (We don't work cheap)
- Time consuming

Dominant Strategies

- Static Analysis
 - IDA Pro
 - Figure out program flow
 - String Searching
 - API Call tracing

Dominant Strategies

- Dynamic Analysis
 - Watch for changes on the system
 - Registry, files, network
 - Monitor System calls at OS/Library level
 - Tools more accessible to unskilled people
 - Sysinternals...

Pros

Static Analysis

- Details
- Precision, full code reversal possible
- Good tools available
- Antivirus
 - It's profitable

Dynamic Analysis

- Fast
- Lower barrier to entry
- High level overview
- Good tools
 - Sysinternals
 - Winalysis
 - CWSandbox

Cons

Static Analysis

- Too much detail
- Full code reversing not necessary
- Tools cumbersome, takes time and training to learn
- Source level analysis full of false positives
- Antivirus
 - Doesn't scale

Dynamic Analysis

- Misses details
- Encourages “next->next->next” analysis
- Tools easily subverted

Which is Better?

- Each perform a task well
- Tools need to be developed on both sides
- Gaps need to be bridged to better integrate the two methods
- This talk is about a partial solution to bridging that gap

Bridging the Gap

- Fundamental problems:
 - Knowing *when* to analyze
 - Data and state change
 - What causes these?
 - Are they important?
 - Automation
 - How do you detect important data changes?

Steps Toward Solutions

- Techniques
 - Debuggers
 - Dynamic Translation
 - Page-fault assisted debugging (Saffron)
 - Sandboxing

Debuggers

- The Good
 - Excellent tools available:
 - OllyDbg, WinDBG, Visual Studio Debugger, gdb
 - APIs
 - PyDbg (thanks Pedram!)
 - Windows Debug API
 - Most have scripting support
 - Allow single-stepping as well as tracing
 - Scripting support allows automation
 - Well tested, vetted, proven

Debuggers (cont.)

- The Bad
 - Detectable!
 - IsDebuggerPresent / MSFT DRM trickery
 - Timing attacks
 - Exception triggering
 - Still too fine-grained
 - Focus is on assembly level
 - Need to integrate with static analysis tools
 - Trivial packer changes confuse debuggers (and us)

What about program flow tracing?

- Visualization should be able to answer questions quickly
- How can we apply this to reverse engineering?
- Aid analyst in understanding program flow

Visualization Strategies

- Program Flow Execution
 - Process Explorer
 - FileMon
- Runtime Instruction Tracing
 - Intel's PIN Framework
 - Allows Analysis of Program Flow
- Call-Graph Tracing
- Basic-Block Visualization

Monitoring Program Execution

- Intel PIN
 - Dynamic instrumentation library
 - Extensible
 - Excellent API
 - Process attach and detach

PIN Basics

NORMAL INSTRUCTIONS

```
pusha  
mov     esi, offset dword_1019000  
lea     edi, [esi-18000h]  
push    edi  
or      ebp, 0FFFFFFFFh  
jmp     short loc_1020332
```


PIN Basics

PIN MODIFIED INSTRUCTIONS

lea edi, [esi-18000h]

push edi

INSERTED PIN PRE-INSTRUCTIONS

INSERTED PIN POST-INSTRUCTIONS

INSERTED PIN PRE-INSTRUCTIONS

INSERTED PIN POST-INSTRUCTIONS

PIN Instruction Monitoring

PIN MODIFIED INSTRUCTIONS

```
pusha
      PIN INSTRUCTIONS
mov    esi, offset dword_1019000
      PIN INSTRUCTIONS
lea    edi, [esi-18000h]
      PIN INSTRUCTIONS
push   edi
      PIN INSTRUCTIONS
or     ebp, 0FFFFFFFh
      PIN INSTRUCTIONS
jmp    short loc_1020332
```


Determine Changed Data

- Look for droppers
- Modified files
- Registry changes
- Unusual withdrawals from bank account
- The Usual Suspects

Implementation

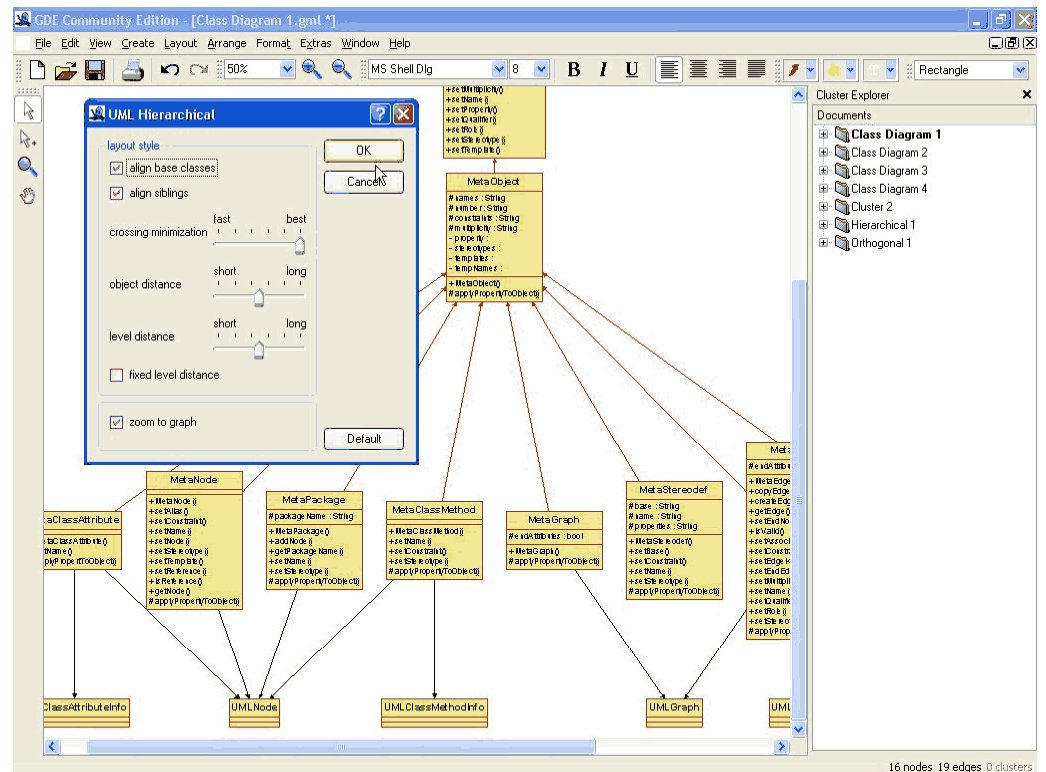
- Setup basic instruction trace
- Monitor via instruction and basic block
- Data Logged:
 - Execution
 - Memory read
 - Memory write

Graph Layout

- Each node (vertex) represents an address of a basic block
- Each line (edge) observed transition
- Collected with custom PIN DLLs
- Processed with Oreas Govisual Diagram Editor (GDE)

Oreas GDE

- Automatic Graph Layout Tool
- Renders large graphs (> 40,000 nodes)
- Automatic layout
 - Tree
 - Circular
 - Symmetric
 - Hierarchical
 - Orthogonal
- <http://www.oreas.com>



Why not Graphviz?

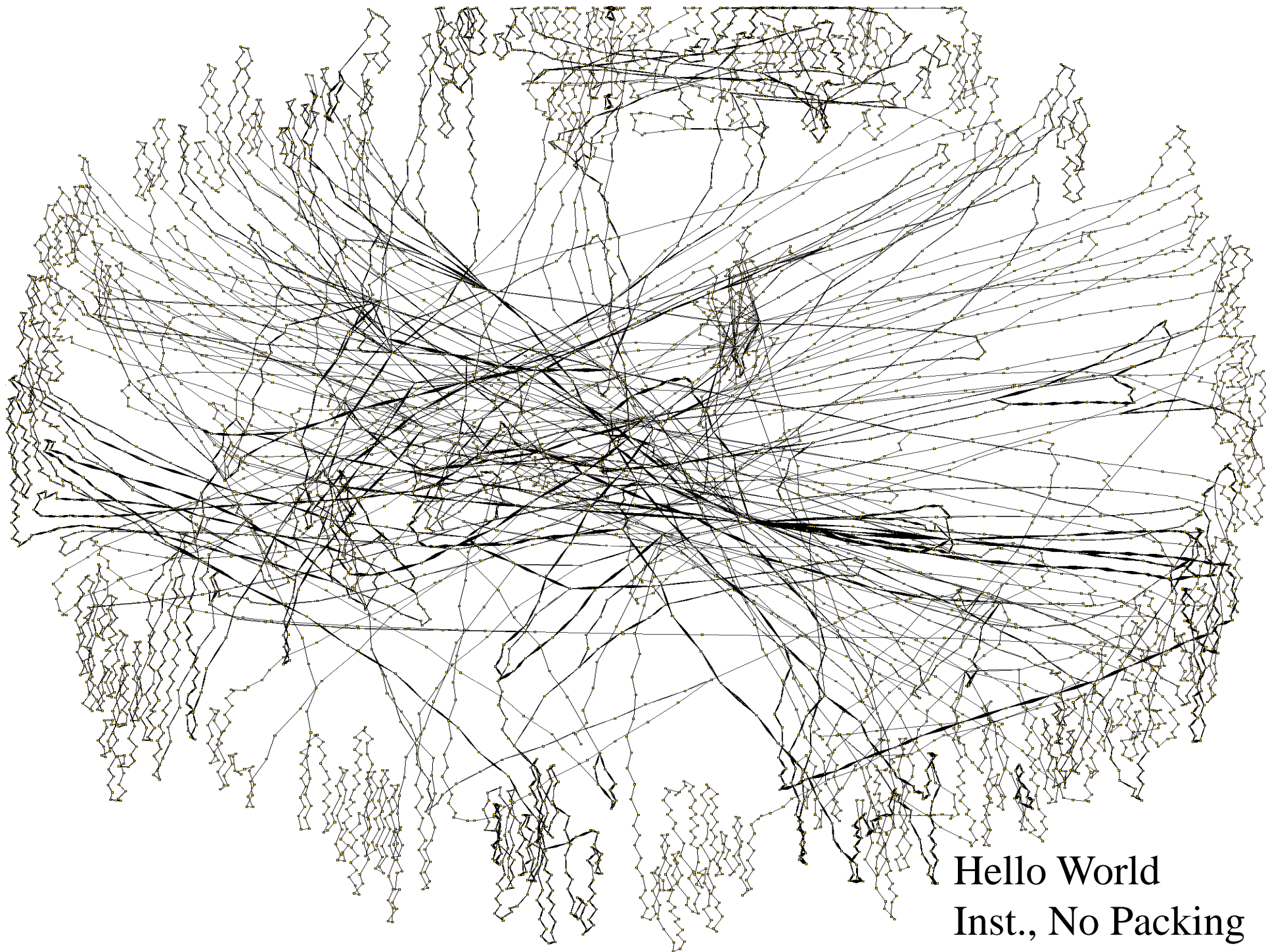
- Fails on extraordinarily large graphs (Memory leaks? Bad algorithm?)
- “Good” for trivial graphs
- Graphs lack visual appeal

Demo: Find the Unpacking Loops

- Simple hello world program

```
int main(int argc, char **argv)
{
    printf("Hello, world\n");
    return 0;
}
```

- First test: Trace each instruction



Hello World
Inst., No Packing

Complexity in Representation

- Previous graph is pretty, but too complex
- Tracing instructions is too complicated
 - This is a small program
 - Moderately complex programs fail
- Data reduction important
 - Compression of instruction graph
 - Tracing at basic block

Basic Blocks

- Set of instructions with branch at the end
- Allows for compaction of tracing data
- More useful analysis information
- Visualization results better

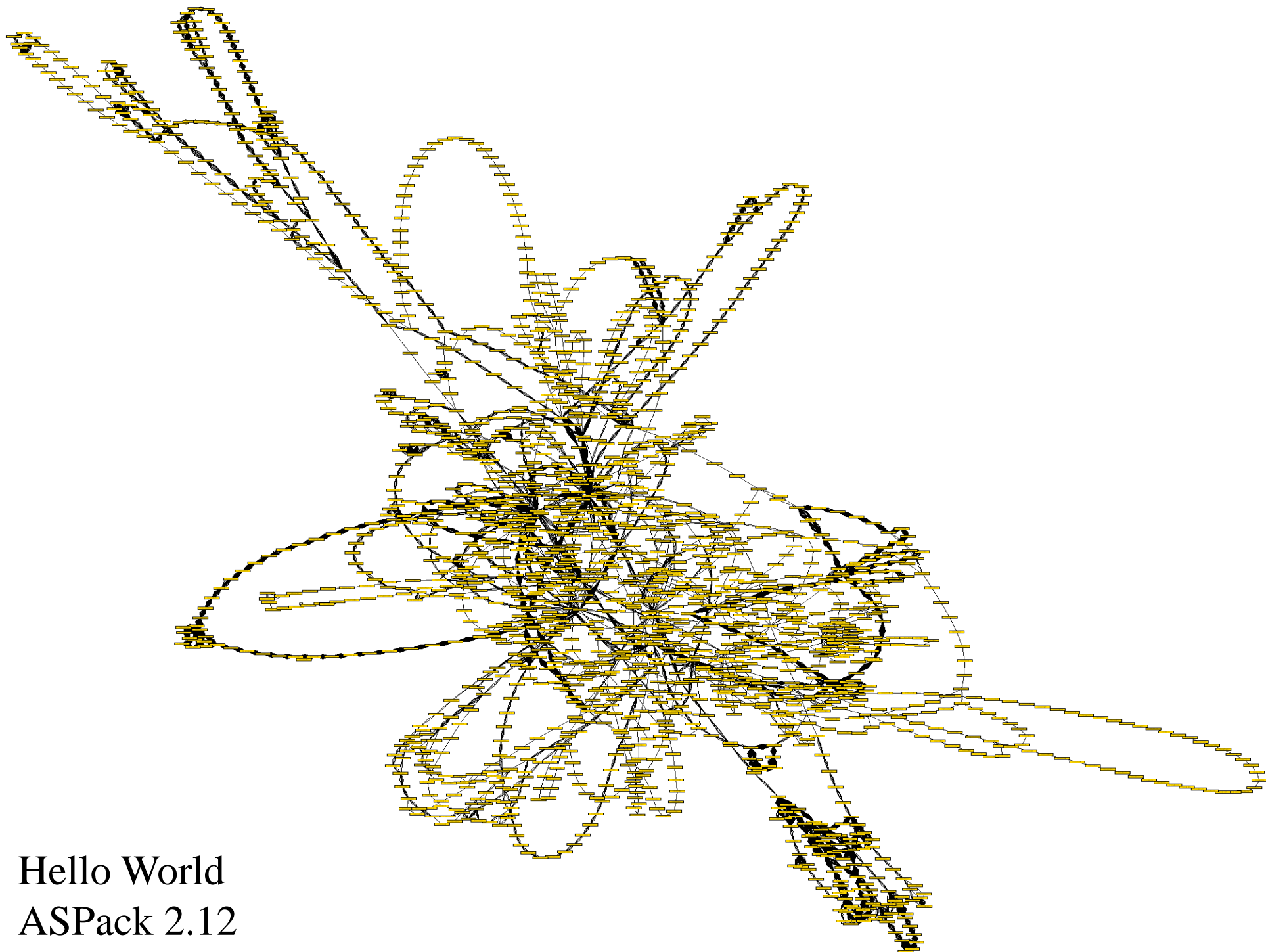




Hello World
Basic Block, No Packing

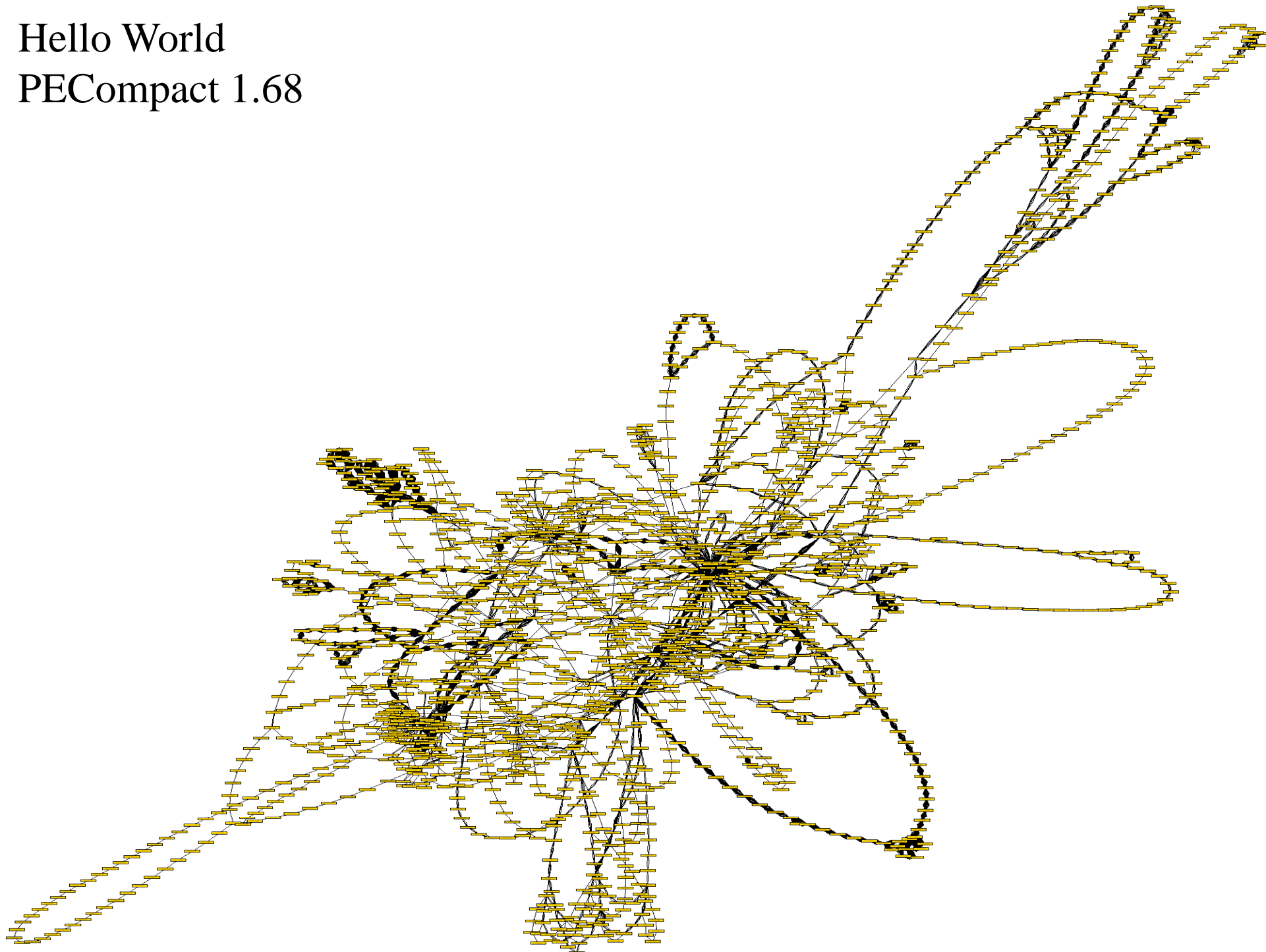
Adding Packers

- Should be able to find the following:
 - Packing loop
 - Main program
- Minimize extraneous information
- Reducing analyst time is the key
- Packers: ASPack, PE Compact, UPX, FSG

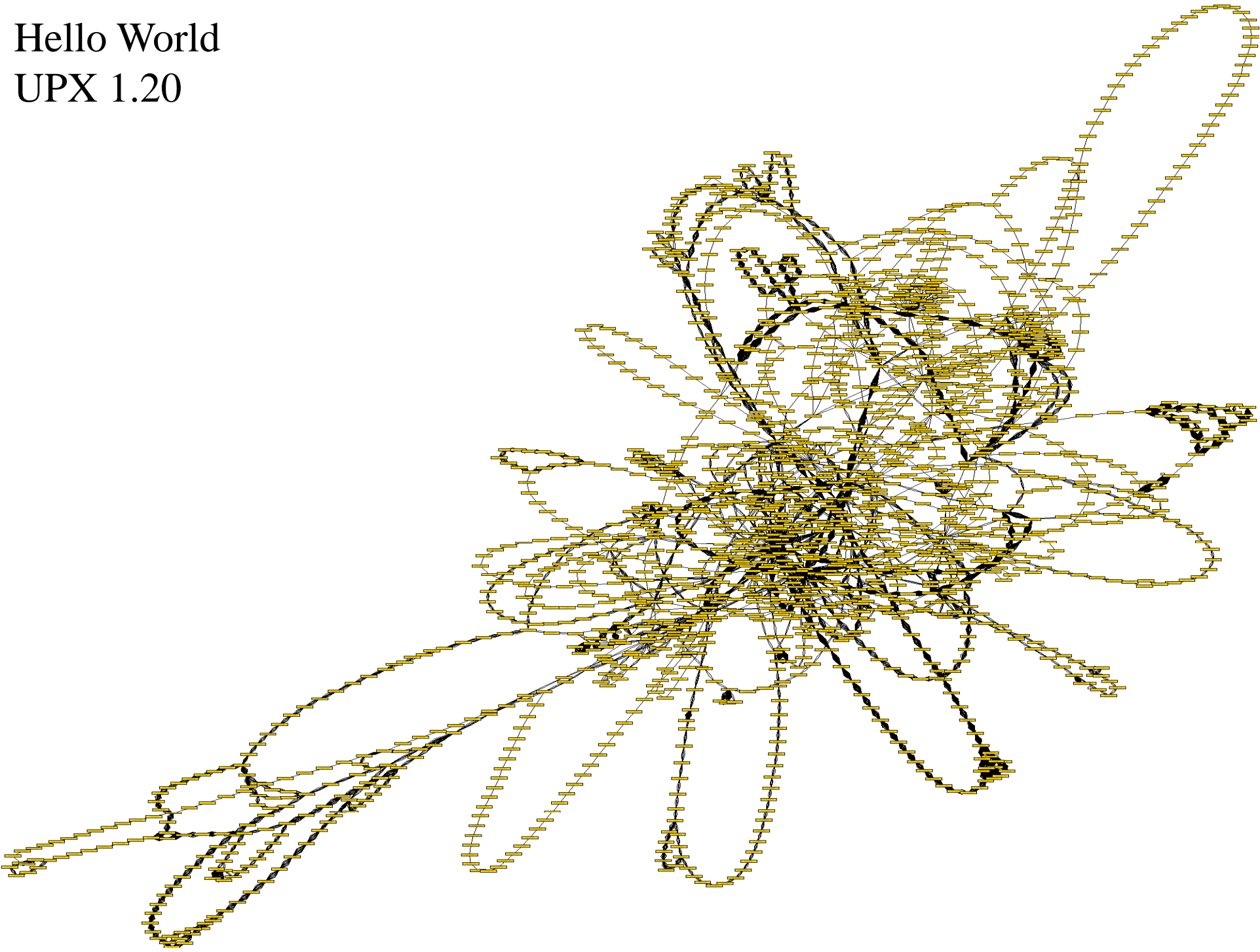


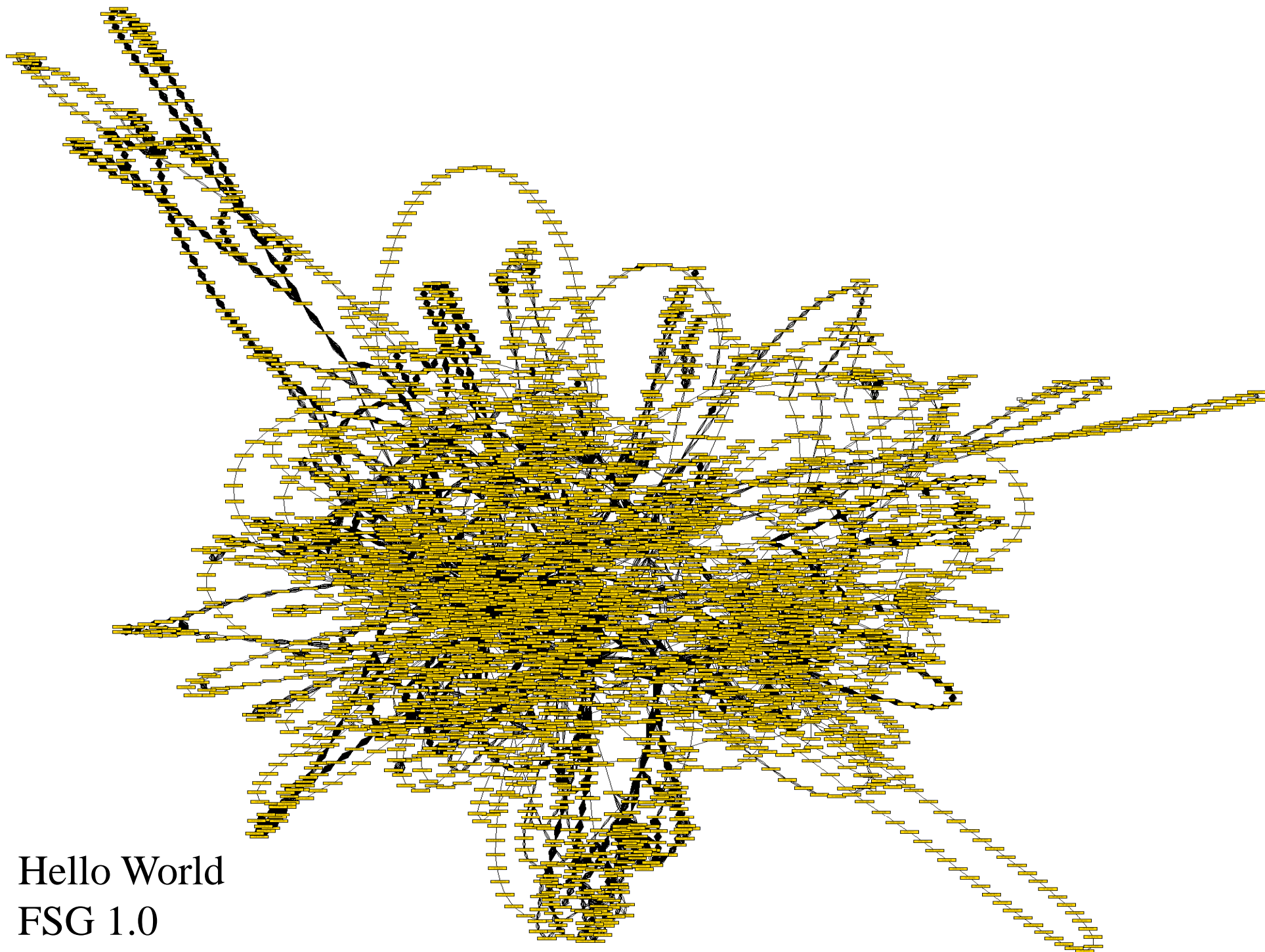
Hello World
ASPack 2.12

Hello World
PECompact 1.68



Hello World
UPX 1.20





Hello World
FSG 1.0

7C9113DC

0040605A

7C918240

00406097

00406037

00406039

00406022

0040601E

0040605F

00406092

0040603E

00406090

00406026

00406029

18238

0040607F

00

00406030

00406041

0040609D

00406086

00406053

00406047

00406045

6082

0040607A

0040609F

11484

0040604F

00406075

00406058

9111FE

Hello World

FSG 1.0

0040606E

C918184

7C911

7C

Visualization Results

- Unpacking loops easy to identify
- Useful for saving analysis time
- Visual appeal
- Narrowing in on relevant data is useful

Memory Checkpointing

- Analyze changes between two execution points of a program
- Useful for comparing differences at certain points
- Entropy analysis of data
- Decryption analysis

Checkpointing

- Determine when to checkpoint
 - Relevant Events
 - Instruction
 - Basic block
 - Page access

Determine Relevant Events

- Malware
 - File write
 - Network communication
 - Any system call
- Commercial Software
 - Most probably decryption point
 - Software load

Relevant Event

- Use system monitoring tools:
 - Filemon, regmon, sysinternals tools
 - Winalysis
 - Wireshark
 - WinDBG, SoftICE, Ollydbg, etc.

Checkpointing

- Preservation of state
 - Register contents
 - Stack contents
 - CPU State
 - **Memory**

What We Care About

- State of memory at a certain event
- Typical checkpoint systems wish to restore
- We want to analyze prior to these events
- Be able to develop a temporal view of program as it changes

Existing Checkpointing Tools

- OS Suspend
- Cryopid
- Memory Paging
- OS Scheduler

Isolating Important Data

- Memory maps
- Memory hotspots
- Entropy Analysis
- Manual exploration

Enabling Analysis

- Prepare for analysis
 - Pausing and suspending execution
 - Debugger
 - Pagefault Debugger (Saffron)
 - System call
 - Copy running process space to disk
 - Reproduce Memory PE view and file PE format
 - Repair Imports

Rebuilding PE files for IDA

How IDA creates its import section .idata and populates subviews Imports, Names

- IMAGE_DIRECTORY_ENTRY_IMPORT
 - RVA (Relative Virtual Address) to Import Directory
- IMAGE_IMPORT_DESCRIPTOR's
 - OriginalFirstThunk
 - RVA to INT (Import Names Table)
 - FirstThunk
 - RVA to IAT (Import Address Table)
- Scan's Code for call's in INT
 - Prepends internal functions to .idata section

Rebuilding PE files for IDA

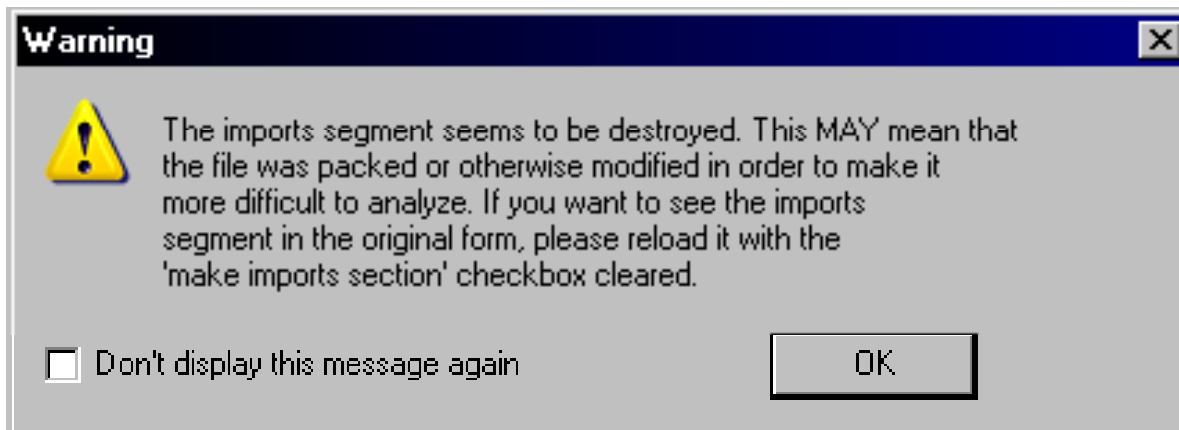
Steps to Recovering INT from packed or encrypted PE

- Unpack using Saffron
 - Discover OEP
- Enumerate Loaded Modules
 - CreateToolhelp32Snapshot, Module32First
- Scan Process heaps for Module Address
 - Translate Virtual Address into RVA
- Rebuild INT and IAT
 - Dump Process memory

Malware Example

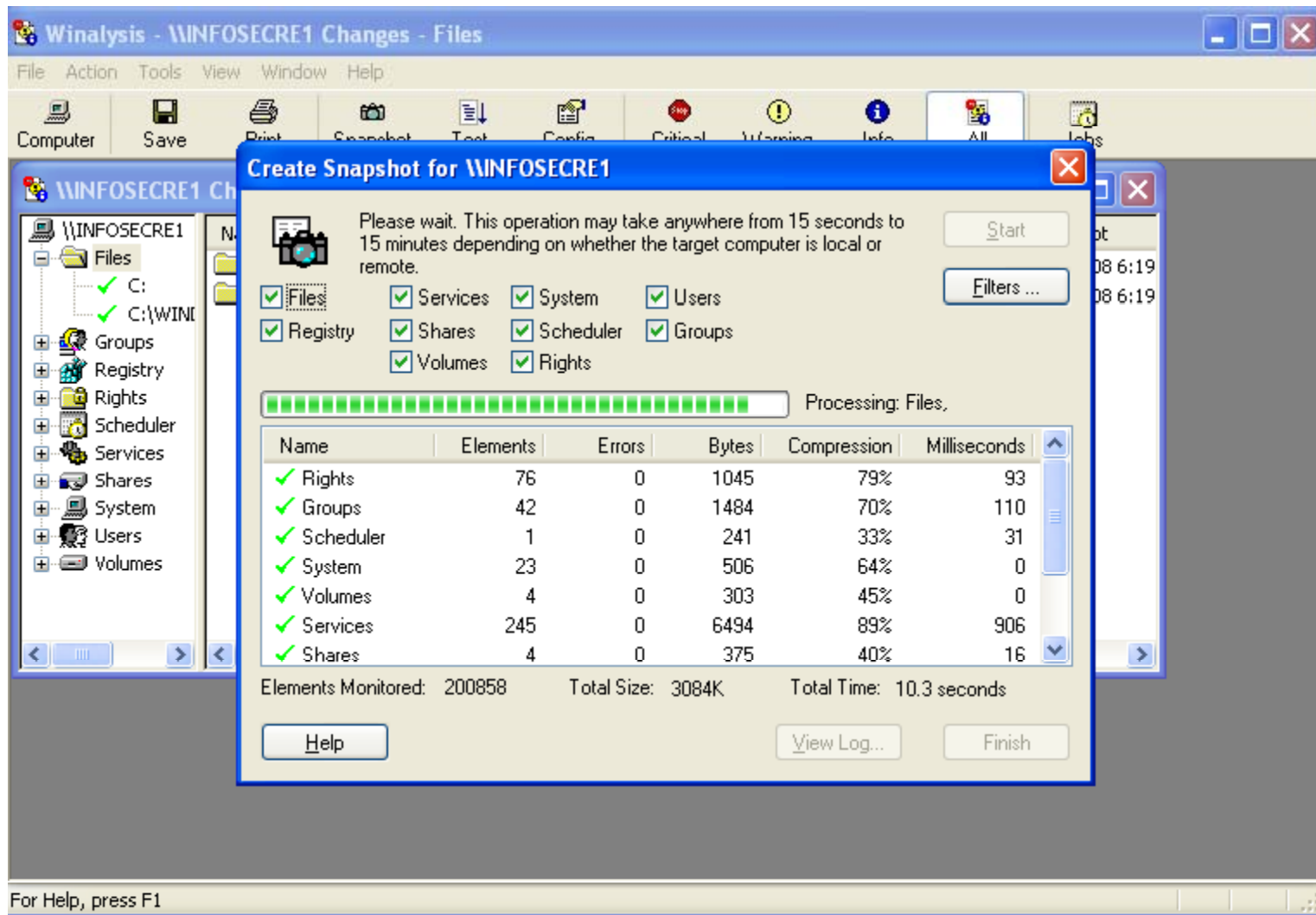
- Start with Storm/Nuwar/Peed Sample
- Found in spam folder e-card.exe
- Analyze using static/dynamic methods
- 591258adc48b422c86730214aef81989
- Download on Offensive Computing

What does IDA Say?



Signs say: Packed with something weird

Winalysis



OC Results

Malware Search

Searching for "591258adc48b422c86730214aef81989"

MD5: 591258adc48b422c86730214aef81989	SHA1: 900d89f8c61055b47a25bff416a97790fd0bce35								
SHA256: 8643b76f3e60477ac714e9d7f0dfbebf4b7e4efd97401be95c305b3220b628cb									
Original Submitted Filename: e-card.exe	Date Added: 2008-03-08 20:53:22.992713								
Magic File Type: MS-DOS executable PE for MS Windows (GUI) Intel 80386 32-bit	Packer Signature: GCC-Win32 / XMINGW [758,3040]								
Anti-Virus Results: <table><tr><td>ClamAV</td><td>Trojan.Peed-130</td></tr><tr><td>BitDefender</td><td>Trojan.Peed.IWX</td></tr><tr><td>AVGScan</td><td>I-Worm/Nuwar.N</td></tr><tr><td>FProt</td><td>W32/StormWorm.gen1</td></tr></table>		ClamAV	Trojan.Peed-130	BitDefender	Trojan.Peed.IWX	AVGScan	I-Worm/Nuwar.N	FProt	W32/StormWorm.gen1
ClamAV	Trojan.Peed-130								
BitDefender	Trojan.Peed.IWX								
AVGScan	I-Worm/Nuwar.N								
FProt	W32/StormWorm.gen1								
Tags: drops f75ced55ddf2005bf949a35534057887 <input type="text"/>	Download Sample Password <i>infected</i>								

Virus Total Results

File **unknown** received on **03.09.2008 10:26:07 (CET)**

Current status: **finished**

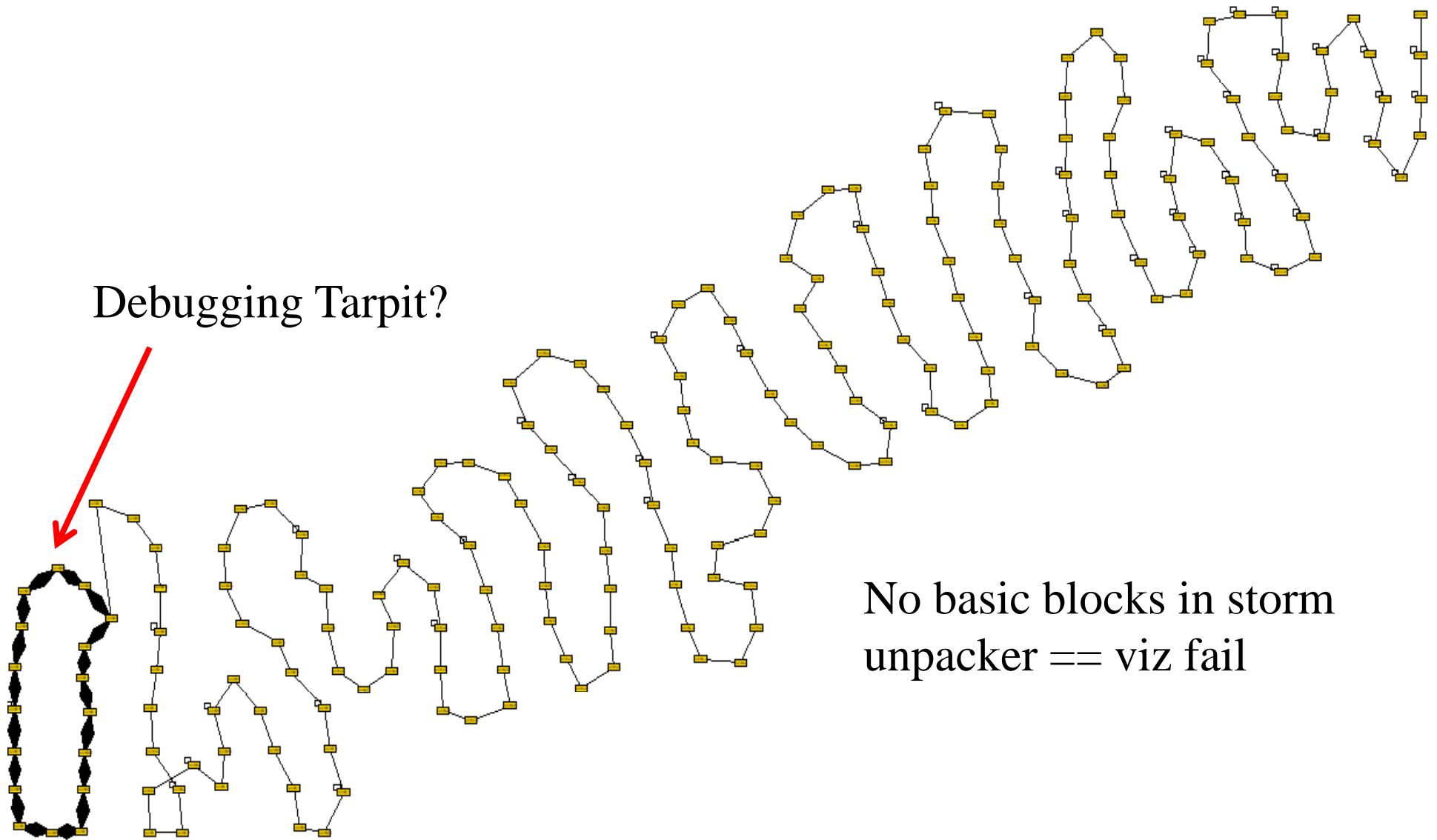
Result: **24/32 (75.00%)**

 [Compact](#)

[Print results](#) 

Antivirus	Version	Last Update	Result
AhnLab-V3	-	-	-
AntiVir	-	-	Worm/Zhelatin.pc
Authentium	-	-	W32/StormWorm.gen1
Avast	-	-	Win32:Zhelatin-CIT
AVG	-	-	I-Worm/Nuwar.N
BitDefender	-	-	Trojan.Peod.IWX
CAT-QuickHeal	-	-	Win32.Packed.Tibs.ic
ClamAV	-	-	Trojan.Peod-130
DrWeb	-	-	Trojan.Packed.357
eSafe	-	-	Suspicious File
eTrust-Vet	-	-	Win32/Sintun!generic
Ewido	-	-	-
F-Prot	-	-	W32/StormWorm.gen1
F-Secure	-	-	Email-Worm.Win32.Zhelatin.vg
FileAdvisor	-	-	-
Fortinet	-	-	W32/PackTibs.M
Ikarus	-	-	Trojan.Peod.IWV
Kaspersky	-	-	Email-Worm.Win32.Zhelatin.vg

Dynamic Analysis

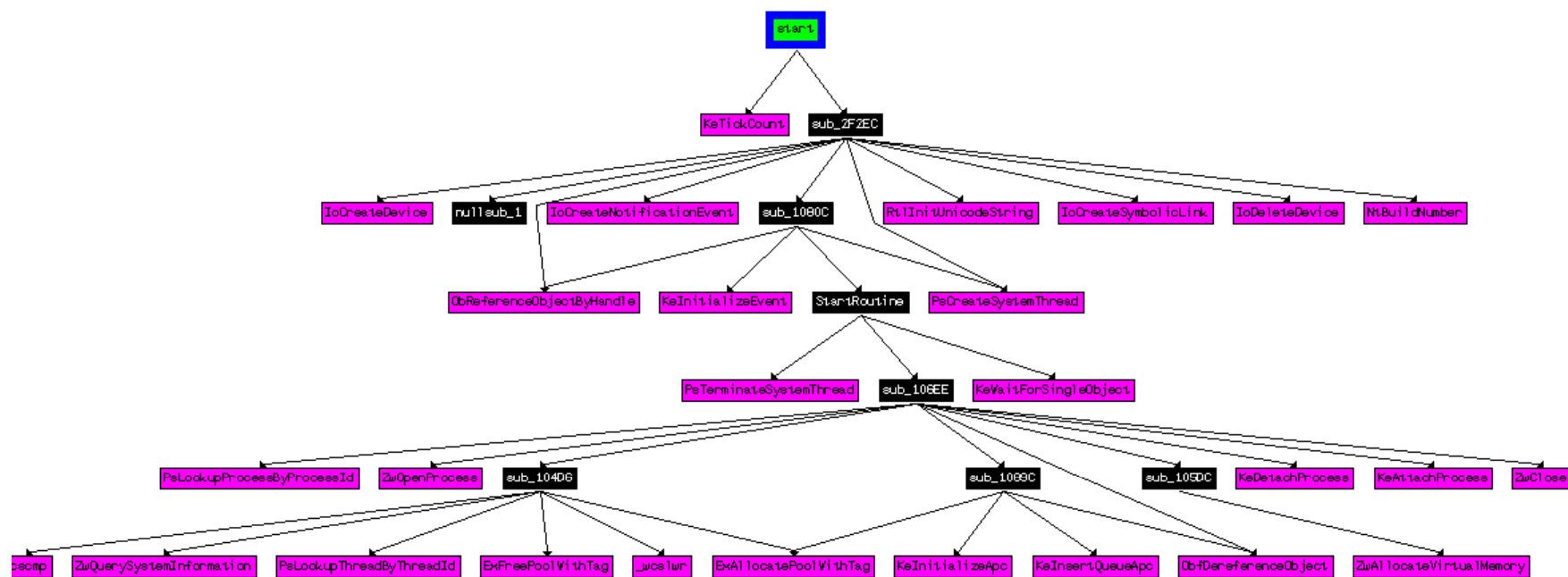


Initial Analysis

- Wireshark shows network traffic
- Debugging reveals that originating source is services.exe
- System call tracing Showed a new file created:
diperto-4417-e33.sys

Diperto-XXXX-xxx.sys

- Device driver loaded by dropper executable
- File is not packed!
- Good disassembly



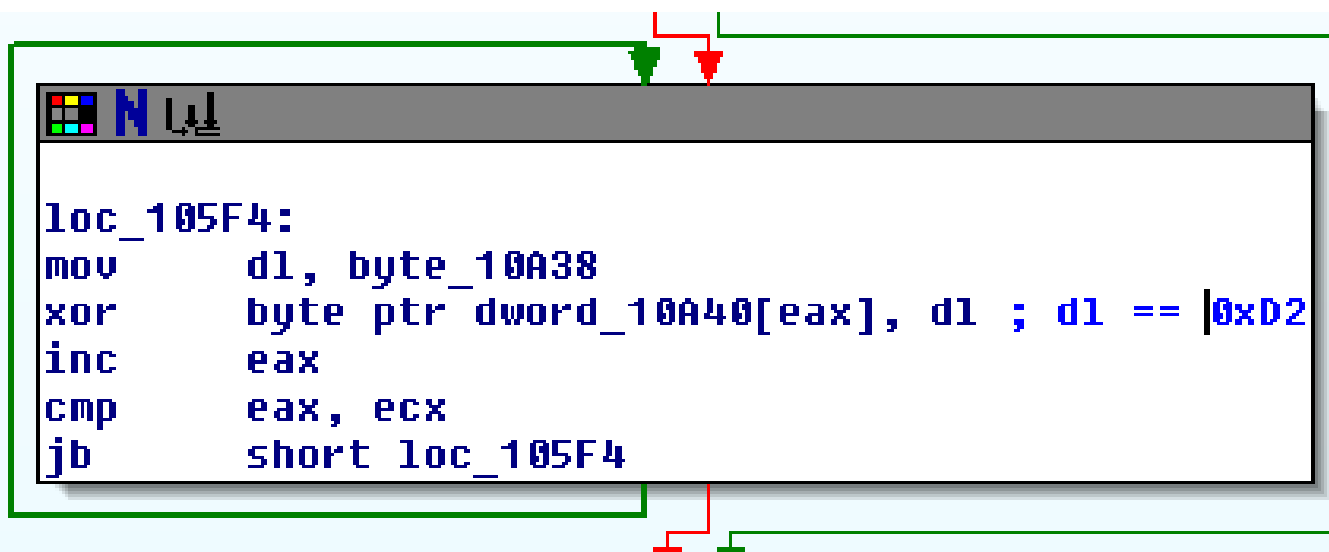
Diperto Features

- Attaches to running process space
- Rootkit finds services.exe
- Injects into the process space

```
push    [ebp+Object]
call    ds:KeAttachProcess
mov     eax, processID
mov     [ebp+ClientId.UniqueProcess], eax
lea     eax, [ebp+ClientId]
push    eax                ; ClientId
lea     eax, [ebp+ObjectAttributes]
push    eax                ; ObjectAttributes
push    1F0FFFh            ; DesiredAccess
lea     eax, [ebp+ProcessHandle]
push    eax                ; ProcessHandle
mov     [ebp+ObjectAttributes.Length], 18h
mov     [ebp+ObjectAttributes.RootDirectory], esi
mov     [ebp+ObjectAttributes.Attributes], 200h
mov     [ebp+ObjectAttributes.ObjectName], esi
mov     [ebp+ObjectAttributes.SecurityDescriptor], esi
mov     [ebp+ObjectAttributes.SecurityQualityOfService], esi
mov     [ebp+ClientId.UniqueThread], esi
call    ds:ZwOpenProcess
cmp     eax, esi
jl      short loc_107B7
```


Diperto Features

- Copies into services.exe address space
- Program runtime actually xor obfuscated

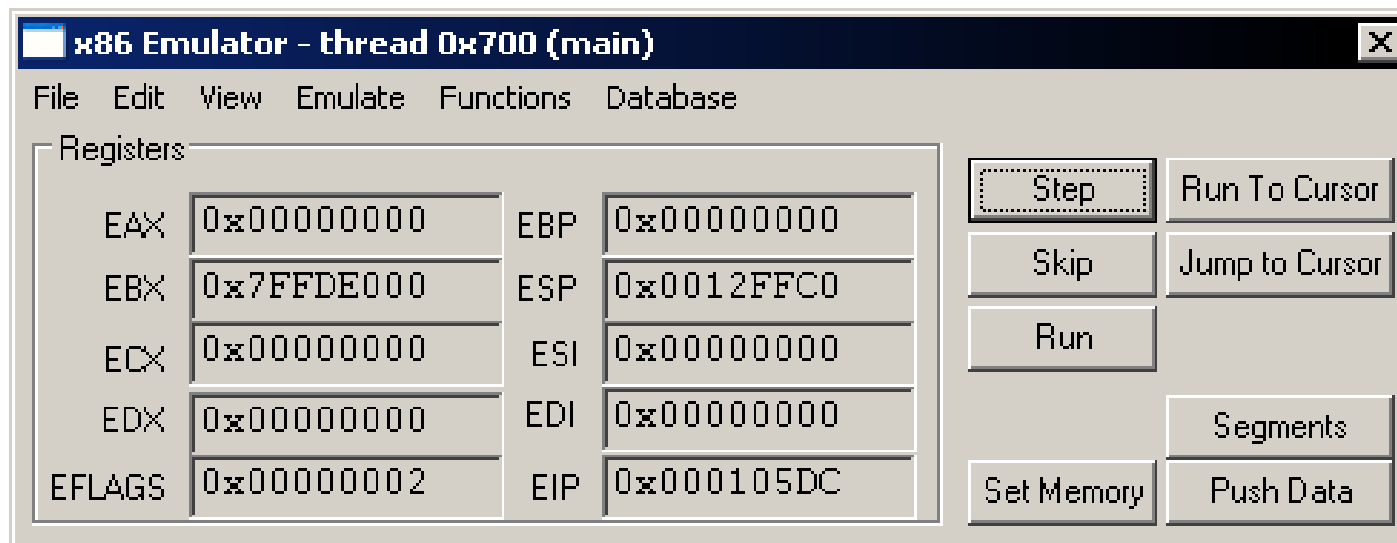


Payload

```
.data:00010A40 dword_10A40      dd  0D242889Fh
.data:00010A40
.data:00010A44      db  0D1h    ; -
.data:00010A45      db  0D2h    ; -
.data:00010A46      db  0D2h    ; -
.data:00010A47      db  0D2h    ; -
.data:00010A48      db  0D6h    ; +
.data:00010A49      db  0D2h    ; -
.data:00010A4A      db  0D2h    ; -
.data:00010A4B      db  0D2h    ; -
.data:00010A4C      db   2Dh    ; -
.data:00010A4D      db   2Dh    ; -
```


Decoding

- Could manually run through decryption
- Could make IDA script
- Better Option: Let it run with x86emu



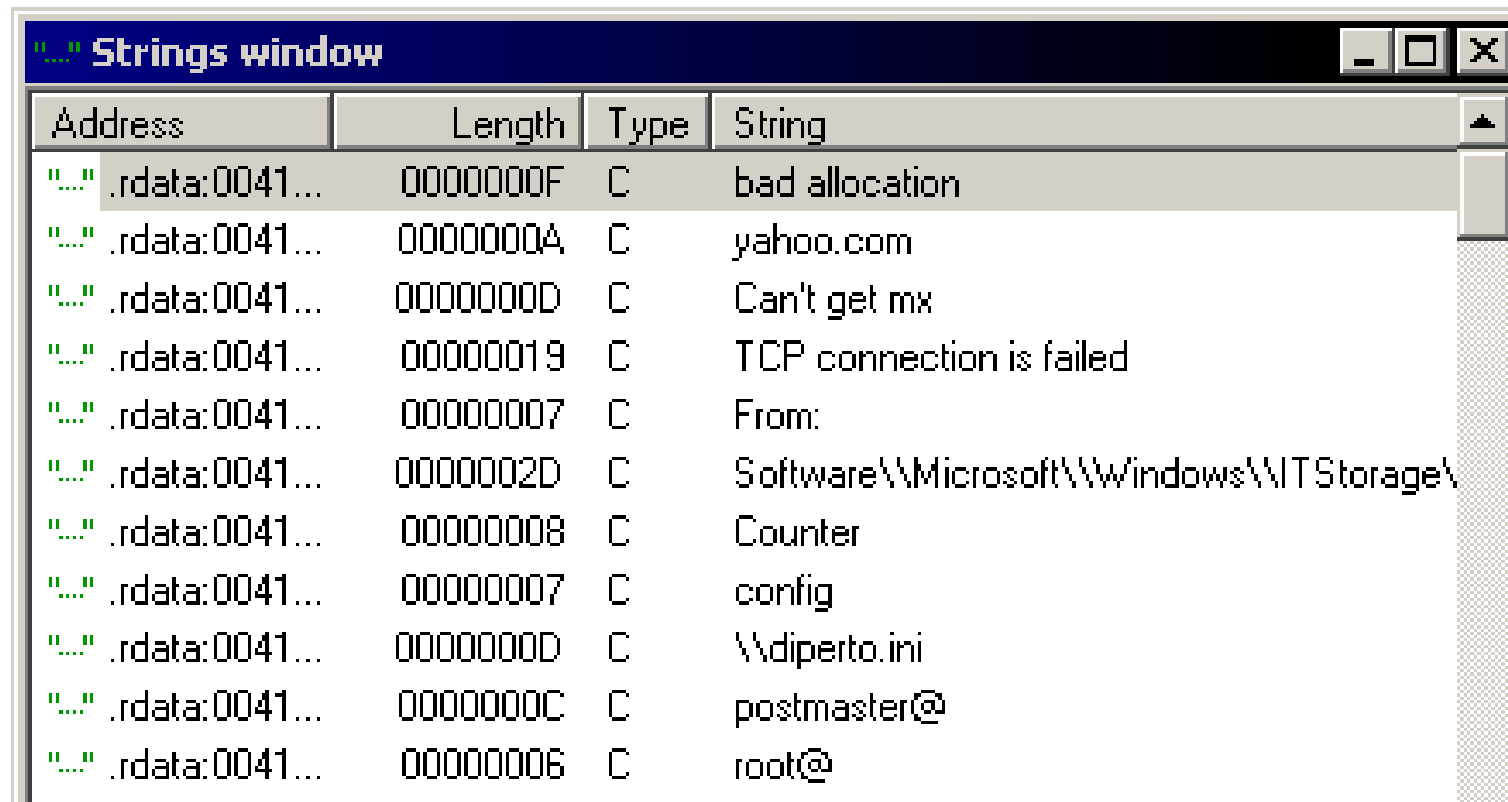
Decoded Data

4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZÉ.■....■.... ..
B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00	+.....@.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 E0 00 00 00a...
0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	■■ ■. ■-!+■L-!Th
69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program canno
74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode.■■■\$......
BC DB F2 CB F8 BA 9C 98 F8 BA 9C 98 F8 BA 9C 98	+ =° £ÿ° £ÿ° £ÿ
F8 BA 9D 98 7C BA 9C 98 DF 7C E7 98 F5 BA 9C 98	° ¥ÿ £ÿ tÿ) £ÿ
FD B6 93 98 FE BA 9C 98 E6 E8 1F 98 F3 BA 9C 98	² ôÿ £ÿµF■ÿ= £ÿ
E6 E8 18 98 C2 BA 9C 98 E6 E8 0D 98 F9 BA 9C 98	µF†ÿ- £ÿµF■ÿ· £ÿ
52 69 63 68 F8 BA 9C 98 00 00 00 00 00 00 00 00	Rich° £ÿ.....

Injected File

- Does the real work of the storm worm
- Has full working unpacked code
- Started via undocumented APC methods
- Starts code running in remote process
- Good obfuscation

Resulting Unpack is Good



A screenshot of a 'Strings window' from a debugger. The window has a title bar with standard Windows controls. Below the title bar is a table with four columns: 'Address', 'Length', 'Type', and 'String'. The table contains ten rows of data, each representing a string found in memory. The first row is highlighted. The strings include 'bad allocation', 'yahoo.com', 'Can't get mx', 'TCP connection is failed', 'From:', a file path, 'Counter', 'config', '\\diperto.ini', 'postmaster@', and 'root@'.

Address	Length	Type	String
"..." rdata:0041...	0000000F	C	bad allocation
"..." rdata:0041...	0000000A	C	yahoo.com
"..." rdata:0041...	0000000D	C	Can't get mx
"..." rdata:0041...	00000019	C	TCP connection is failed
"..." rdata:0041...	00000007	C	From:
"..." rdata:0041...	0000002D	C	Software\\Microsoft\\Windows\\IT Storage\\
"..." rdata:0041...	00000008	C	Counter
"..." rdata:0041...	00000007	C	config
"..." rdata:0041...	0000000D	C	\\diperto.ini
"..." rdata:0041...	0000000C	C	postmaster@
"..." rdata:0041...	00000006	C	root@

Conclusion

- Dynamic Runtime Visualization shows process change over time
- Multiple checkpoints allow for analysis over multiple program states
- Leverage existing tools with time dependant data

References

- Visualization Grand Challenges: Illuminating the Path
http://nvac.pnl.gov/docs/RD_Agenda_NVAC_chapter1.pdf
- Dynamic Data Visualization of Meteorological Data
ASA-JSM Data Exposition, 2006
- Visual Signatures in Video Visualization
IEEE Transactions on Visualization and Computer Graphics, Vol.12, No. 5, September/October 2006
- Static Visualization of Dynamic Data Flow Visual Program Execution
Proceedings of the Sixth International Conference on Information Visualization, IV 2002
- Hoglound, G., McGraw, G., Exploiting Software: How to Break Code, *Chapter 3, Addison Wesley, 2004*
- Amini, P., Process Stalker, *OpenRCE*, http://pedram.redhive.com/code/process_stalker/
- Amini, P., PaiMei, *OpenRCE* <http://www.openrce.org/downloads/details/208/PaiMei>
- Eagle, C., x86emu, <http://ida-x86emu.sourceforge.net/>
- C. Luck, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V.J. Reddi, K. Hazelwood, Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation, *Proceedings of the 2005 Conference on Programming and Language Design and Implementation*, 2005
- Oreas GDE, http://www.oreas.com/index_en.php

**Latest slides and code can be found on
offensivecomputing.net**