Mario Vuksan & Tomislav Peričin, ReversingLabs Corporation

FAST & FURIOUS REVERSE ENGINEERING WITH TITANENGINE
Agenda

- Obligatory Scare Talk
- Why should you care?
- What is the problem?
- How can TitanEngine change the world?
- Show ME!
- Show ME!
- Show ME!
- How can I help?

ReversingLabs Corporation
Fighting Malware: Old Problem

Inadequate Infrastructure: New Problem
Exponential Growth in Malware
YIELDS
Exponential Growth in Signatures

Signatures (mil)

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7

2004 2005 2006 2007 2008

5,490,960+

ReversingLabs Corporation
DEMANDING
Malware Wars
Army of Threat Researchers
RESULTING IN
Denial of Service on Threat Response Teams
So What?
Security Industry is a For-Profit Entity
We’ll Simply Hire More Bodies
But Could We Get Enough Bodies?
Can’t Hire Enough?
Combine those we have into one Worldwide Non-profit Entity
(Bwa-ha-ha!)
OR...

We could simply overload them...
Is an overloaded anti-malware analyst an asset or a liability?
Henry Ford

- Anti-Malware labs are factories
- 100-200+ Analyst teams
- Advanced workflows
- Multiple levels of management
- Modern labor laws apply: No 20+ hour days
- Productivity can be improved
- Work process can be studied
- Improvements *COULD* be devised…

ReversingLabs Corporation
So how can Labs do more?

- Charge more, Hire more
- Invest in automation, Invest in heuristics
- Deploy proactive modules, Buy competitors
- All the usual stuff
- ... and they could revise their processes
So how can Labs do more?

- 1,000s of OllyDBG and IDAPro scripts can better be reused; could be generalized
- Sample analysis, OEP discovery could benefit all team members
- Reversing should be a team effort
We have to do it better...
Competition is tough

- Bad guys
  - Rise of $$ motivated custom attacks
  - Resourceful crime syndicates
Protection is lacking

- Signatures only “important” for threats
- Need for other types of protection
- Behavioral & HIPS tools *that work*
Yet manual analysis is still the only certain bet!
Passion for binary protection

- Meatiest task today is dealing with protection techniques
- Task repetition, Error prone, Not reusable
- Large number of file formats can be infected and used for malware
Passion for binary protection

- Executable files == most significant threat
- Executables == the “usual suspect” for malware
  - 85% of malware samples are packed
  - Packing hides malware, hardens its detection
- Packed or protected doesn’t mean bad!
  - 10% of legitimate software is packed
Passion for binary protection

- Legit use for packers & protectors:
  - Compressed binaries decrease bandwidth usage
  - Protect intellectual property
  - Protect from code theft
  - Anti-tampering in multi-player games
  - Safeguard licensing code

- Successfully used by malware authors
  - For all the same reasons
Analyzing Malware

- Malware File Analysis Requires:
  - In-depth knowledge of how PE works
  - In-depth knowledge of how Windows works
  - Various tools to make you reach your goal

- Understanding of Basic Shell Divisions:
  - Packers, Protectors, Crypters, Bundlers & Hybrids
  - Custom malware-specific packers & protectors
What’s the Reversing Process Today?
Reversing in action

Today

- Inspect the Sample
  - Identify the packing shell or compiler

![PEiD Interface]

PEiD

ReversingLabs Corporation
Reversing in action

- Unpack the Sample
  - Execute it to the original entry point
Reversing in action

- Unpack the Sample
  - Execute it to the original entry point
Reversing in action

- Unpack the Sample
  - Execute it to the original entry point
Reversing in action

Today

- Unpack the Sample
  - Dump the process memory
Reversing in action | Today

- Unpack the Sample
  - Fix the import table
Problems with File analysis

- File analysis takes time
  - Identifying requires keeping up with shells
  - Shells evolve & have different forms
- Analysts get more samples than they can handle
- File unpacking takes even more time
  - Protection “tricks” continue to evolve
  - Yet, this process can be automated!
Fast Reversing | Tomorrow

- TitanEngine key features:
  - Framework designed to work with PE files
  - 250 documented functions
  - Easy automation of all reversing tools
  - Supports both x86 and x64
  - Can create:
    - Static, Dynamic & Generic unpackers
    - New file analysis tools
  - Tested on over 150 unpackers
  - It's free and open source!
Furious Reversing | Tomorrow

- Engine simulates reverse engineer’s presence
  - Unpacking process has the same steps:
    - Debugs until entry point
    - Dumps memory to disk
    - Collects data for import fixing
    - Collects data for relocation fixing
    - Custom fixes (Code splices, Entry point, ...)

ReversingLabs Corporation
TitanEngine | Content

- SDK Contains:
  - Integrated x86/x64 debugger
  - Integrated x86/x64 disassembler
  - Integrated memory dumper
  - Integrated import tracer & fixer
  - Integrated relocation fixer
  - Integrated file realigner
  - TLS, Resources, Exports...
TitanEngine | Debugger

- Integrated x86/x64 Debugger
  - Attach / Detach
  - Trace, including single stepping
  - Set several types of breakpoints:
    - Software (INT3)
    - Hardware
    - Memory
    - Flexible
    - API
  - Access debugged file’s context
TitanEngine Debugger

- Integrated x86/x64 Debugger
  - Disassembly instructions
    - Disassemble a length
    - Full disassemble
  - Memory manipulation
    - Find, Replace, Patch, Fill...
  - Get call/jump destination
  - Check if the jump will execute or not
  - Thread module for thread manipulation
  - Librarian module for module manipulation
TitanEngine | Dumper

- Integrated Memory Dumper
  - Dump memory
    - Process, regions or modules
  - Paste PE header from disk to memory
  - Manipulate file sections
    - Extract, resort, add, delete & resize
  - Manipulate file overlay
    - Find, extract, add, copy & remove
TitanEngine | Dumper

- Integrated Memory Dumper
  - Convert addresses
    - From relative to physical, and vice-versa
    - Get section number from address
  - PE header data
    - Get and set PE header values
TitanEngine | Importer

- Integrated Import Fixer
  - Build new import tables on the fly
  - Get API information
    - API address in both your & debugged process
    - DLL to hold API from API address
    - Remote & local DLL loaded base
    - API name from address
    - API Forwarders

ReversingLabs Corporation
TitanEngine | Importer

- Integrated Import Fixer
  - Automatic import table functions:
    - Locate import table in the memory
    - Fix the import table automatically
    - Fix import eliminations, automatically
  - Enumerate and handle import table data
  - Move import table from one file to another
  - Load import table from any PE file
TitanEngine | Tracer

- Integrated Import Tracer
  - Identify import redirections and eliminations
    - Fix known import protections
  - Use integrated tracers to resolve imports
    - Static disassembly tracer
    - Static hasher disassembly tracer
  - Use ImpRec modules to fix redirections
TitanEngine | Relocater

- Integrated Relocation Fixer
  - Build new relocation table on the fly
  - Resolve relocation table
    - Grab relocation table directly from the process
    - Make & compare memory snapshots
  - Remove relocation table from the file
  - Relocate file to new image base
TitanEngine | Realigner

- Integrated File Realigner
  - Validate PE files
  - Fix broken PE files
  - Realign files: reduce size & validate
  - Fix header checksum
  - Wipe sections
TitanEngine | The Rest...

- TLS
  - Remove callbacks
  - Break at callbacks
- Exporter
  - Build export tables on the fly
- Handler
  - Close remote handles
  - Get file lock handles
  - Find open mutexes
TitanEngine | The Rest...

- Resource
  - Extract resource
- Remote
  - Load & Free libraries into running process
- OEP Finder
  - Get OEP location generically
- Static
  - Unpack files statically
Back to Basics: Shell Modifier Types

- Shell Division
  - Crypters
  - Packers
  - Protectors
  - Bundlers
    - Data bundlers
    - Overlay/Resource bundlers
  - Hybrids
Packed File Layout

Packed file layout

- DOS
- PE
- Sections
- STUB
- Overlay

Execution layout

- Internal data decompression
- Section decompression
- Relocation to new base
- Import resolving
- TLS callback emulation
- Entry point jump
Unpacker Types...

- Basic Unpacker Division
  - Static unpackers:
    - **Pro:** simple, fast & supported by TitanEngine
    - **Con:** don’t work if internal shell mechanisms change
  - Dynamic unpackers:
    - **Pro:** “simple”, fast & supported by TitanEngine
    - **Con:** carry a certain risk of file execution!
  - Generic unpackers:
    - **Pro:** Can support large number of similar shells
    - **Con:** Can be highly inaccurate!

ReversingLabs Corporation
Writing an Unpacker...

- Analyze the Packing Shell
  - **Step 1**
    - Determine protection types
      - Design ways to avoid them
      - Determine method to resolve custom protections
      - Determine method to skip entry point layer protection
      - Determine if we can automate file identification
Writing an Unpacker...

- Analyze the Packing Shell
  - **Step 2**
    - Locate packing shell’s important parts
      - Where does it fill import table?
      - Where does it relocate the file?
      - How does it jump to OEP?
    - Identify byte patterns, using *lots* of samples!
      - Proper patterns contain wild cards
      - Proper patterns work on all samples
      - Proper patterns are based on *multiple* compiler cases!
Writing an Unpacker...

- Writing the Unpacking Code
  - **Step 3**
    - Select the best platform for unpacker creation
      - Select framework
        - Write a custom one, or select existing
      - Select programming language
  - **Step 4**
    - Write and test it
      - Test on as many samples as you can get your hands on!
Dynamic Unpacker Layout

- GetPEdata
  - InitDebug
  - SetBPX
  - DebugLoop
  - StopDebug

- Import fix
  - Reloc fix
  - Entry point

Import Data Gathering

- LoadLibrary
- GetProcAddress
- Packer segment

- ImporterAddNewDll
- ImporterAddNewAPI
Creating a Dynamic Unpacker for UPX:

- Gathering info on the packer
  - Free & open source
  - Can pack DLL & EXE files
  - Multiple platforms supported
  - DEP supported but no x64 support
- Multiple unpackers exist
  - UPX can decompress itself!
- Multiple signatures available
UPX | Analysis

- Packer Code Points of Interest
  - Point of interest #1:
    - Import table filling (string case)

```
```

Bytes: 50 83 C7 08 FF 96 4C 85 00 00

BPX
UPX | Analysis

- Packer Code Points of Interest
  - Point of interest #1:
    - Import table filling (ordinal case)

```
/*40C304*/  MOVZX EAX,WORD PTR DS:[EDI] /*40C307*/  INC EDI /*40C308*/  PUSH EAX /*40C309*/  INC EDI /*40C30A*/  DB B9 /*40C30B*/  PUSH EDI /*40C30C*/  DEC EAX /*40C30D*/  REPNE SCAS BYTE PTR ES:[EDI] /*40C30E*/  CALL NEAR DWORD PTR DS:[ESI+CBF8] /*40C30F*/  OR EAX,EAX
```

Bytes:
```
50 47 ?? 57 48 F2 AE (BPX)
```

Bytes:
```
57 48 F2 AE ?? FF 96 F8 CB 00 00
```

↑

BPX
UPX | Analysis

- Packer Code Points of Interest
  - Point of interest #2:
    - Relocating file to loaded base

```
/*3D2C4A*/  ADD EDI,4
/*3D2C4D*/  LEA EBX, DWORD PTR DS:[ESI-4]
/*3D2C50*/  XOR EAX, EAX
/*3D2C52*/  MOV AL, BYTE PTR DS:[EDI]
/*3D2C54*/  INC EDI
/*3D2C55*/  OR EAX, EAX
/*3D2C57*/  JE SHORT iPakege.003D2C7B
/*3D2C59*/  CMP AL, 0EF
/*3D2C5B*/  JA SHORT iPakege.003D2C6E
/*3D2C5D*/  ADD EBX, EAX
/*3D2C5F*/  MOV EAX, DWORD PTR DS:[EBX]
/*3D2C61*/  XCHG AH, AL
/*3D2C63*/  ROL EAX, 10
/*3D2C66*/  XCHG AH, AL
/*3D2C68*/  ADD EAX, ESI
/*3D2C6A*/  MOV DWORD PTR DS:[EBX], EAX
/*3D2C6C*/  JMP SHORT iPakcage.003D2C50
/*3D2C6E*/  AND AL, 8F
/*3D2C70*/  SHL EAX, 10
/*3D2C73*/  MOV AX, WORD PTR DS:[EDI]
/*3D2C76*/  ADD EDI, 2
/*3D2C79*/  JMP SHORT iPakcage.003D2C5D
```
UPX | Analysis

- Packer Code Points of Interest
  - Point of interest #3:
    - Entry point jump (old method)

```assembly
/*4082A1*/  MOV DWORD PTR DS:[EBX],EAX
/*4082A3*/  ADD EBX,4
/*4082A6*/  JMP SHORT crackme_.00408289
/*4082A8*/  CALL NEAR DWORD PTR DS:[ESI+8554]
/*4082AE*/  POPAD
/*4082AF*/  JMP crackme_.004012C0
```

**Bytes:**

```
61 E9 0C 90 FF FF
```

BPX

ReversingLabs Corporation
UPX | Analysis

- **Packer Code Points of Interest**
  - **Point of interest #3:**
    - Entry point jump (new method)

```assembly
/*45F5F5*/     POPAD
/*45F5F6*/     LEA EAX,DWORD PTR SS:[ESP-80]
/*45F5FA*/     PUSH 0
/*45F5FC*/     CMP ESP,EAX
/*45F5FE*/     JNZ SHORT dELPHI_u.0045F5FA
/*45F600*/     SUB ESP,-80
/*45F603*/     JMP dELPHI_u.0044CF38
```

**Bytes:**

```
83 EC ?? E9 30 D9 FE FF
```

BPX
UPX | Unpacker

- Starting the “Engine”
  - Read interesting file data
    - ImageBase, AddressOfEntryPoint, ...
  - Initialize the debugger
    - InitDebugEx for executables
    - InitDLLDebug for libraries
  - Set initial breakpoint at packer EP
  - DebugLoop();
UPX | Unpacker EP Callback

- Finding Our Points of Interest
  - Find import filling code
    - Set breakpoints pointing to import handle code
      - There are one or two breakpoints here
  - Find “relocate to new base” code
    - Set breakpoints pointing to snapshot code
      - There is one breakpoint here (optional)
  - Find entry point jump
    - Set breakpoints pointing to unpack finalization
      - There is one breakpoint here (but two patterns!)
UPX | Unpacker Breakpoints

- Assign Callbacks to Our Breakpoints
  - Import fixing callback
    - Breakpoint #1; Loading new library
      - In this callback call ImporterAddNewDLL
      - Data: EAX holds the pointer to string in remote process
UPX | Unpacker Breakpoints

- Assign Callbacks to Our Breakpoints
  - Import fixing callback
    - Breakpoint #2: Getting API address (string case)
      - In this callback call ImporterAddNewAPI
      - Data: EAX holds the pointer to string in *remote process*
      - Data: EBX holds the data write address
    - Breakpoint #3: Getting API address (ordinal case)
      - In this callback call ImporterAddNewAPI
      - Data: EDI holds the ordinal number
      - Data: EBX holds the data write address
Assign Callbacks to Our Breakpoints
  - Relocation fixing callback
    - Breakpoint #4; Snapshot #1
      - This is optional breakpoint, present only if file is DLL
      - In this callback we create a snapshot file
      - Function RelocaterMakeSnapshoot
        - Memory which will be snapshot is first PE section
UPX | Unpacker Breakpoints

- Assign Callbacks to Our Breakpoints
  - Original entry point callback
    - Breakpoint #5
      - Dump the process with DumpProcess
      - Fix (possibly broken) PE header with PastePEHeader
      - Make second relocation snapshot & compare them
      - Add new section for IAT and export IAT to it
        - ImporterExportIAT
      - Add new section for relocations and export them
        - RelocaterExportRelocation / RelocaterChangeFileBase
      - Realign the file with RealignPE
      - Move overlay from original to unpacked file
      - StopDebug();
UPX  |  DEMO

- DEMO - UPX Unpacker
  - But does it actually work?
File -> New Unpacker...

- Create a Generic *Executable* Unpacker
  - No signatures, no patterns, no problem...
    - Generically determine OEP location
    - EP can not be fixed without getting into specifics
    - Automatically fix imports
      - Fix redirections & import eliminations
    - No hassle with relocations
      - But generic DLL unpacker is possible!
    - Dual process dilemma?
Generic OEP finder blueprint

- Creating a generic entry point finder

1. EIP inside sections
2. Already written to
3. Hash has changed
4. Not a simple redirection
Generic Unpacker | DEMO

- RL!dePacker 2.0
  - But does it actually work?
AlexProtector | DEMO

- ImportStudio 2.0
  - Tool similar to ImpRec used to fix imports
  - Demo: fixing import eliminations
ImportStudio 2.0
- Tool similar to ImpRec used to fix imports
- Demo: using ImpRec plugins
TitanEngine | What’s Next?

- Extend Framework
  - File function analysis
  - Plugins, modules and scripts
  - Integrated file identification
  - Extend SDK to Delphi and MASM
  - Extend SDK to python and ruby

- More Samples of Usage
  - One unpacker per week project

- More Analysis Tools Built Around It
  - UnpackStudio, MFK...
TitanEngine – How to Help?

- [http://titan.reversinglabs.com](http://titan.reversinglabs.com)
- Open Source Project
- Contribute Solutions
- Help others with tutorials
- Contribute Code
- Forums
Questions?

Questions?
(What Would You Like to Know)