

Owning Antivirus

Alex Wheeler (alexbling@gmail.com)

Neel Mehta
(nmehta@iss.net)

Why AV?

Attractive Attack Surface

- Gateways, Servers, Clients, ISP's, Third Party Vendor Products
- Heterogeneous and Layered Environments

Why AV?

–Un-trusted Data Processing

- Must be reachable by external input to be useful

Why AV?

- Run on a variety of Platforms
 - Windows, Linux, Solaris, Mac

How Does AV work?

- Signature vs. Behavior
 - Pattern-matching / regex
 - File-format decomposition

How Does AV work?

- Enterprise vs. Consumer Architecture
 - \$\$\$ & Manageability

How Does AV work?

- Common Core Components
 - IO filters
 - Format Engines

How Does AV work?

– Standard Features

- Updates
- Multi-Threat detection

How Does AV work?

– Common Configurations

- Scan level
- Scan sizes
- Scan Method

Code Coverage - Signatures

– Field Values

- Max Len (eg. ARJ header len 0xa28)
- Magic (eg. PECOFF – “MZ” & “PE”)

– Field Sizes

- PE Section Header 0x28
- Tar Object 0x200

– Strings

- PECOFF – section labels, common libraries

– Ida Examples

- LHA
- ARJ
- UPX

Code Coverage – Core Utilities

– Read

- Easy to spot
- Closest audit point to un-trusted input
- Usually wrapped & buffered
- Some truncate length

Code Coverage – Core Utilities

– Allocation

- Any calculations to length are interesting
- Usually wrapped
- Some check 4 zero
- Some add to length for internal headers
- Some wrappers will truncate length

Code Coverage –Constructs

– Conversions

- String/Number
- Byte Ordering

Code Coverage –Constructs

- Checksum, CRC, etc.
 - Easy to spot (ror, xor, etc. in a loop)
 - Gives un-trusted input context

Code Coverage –Constructs

- Inherited File Structures & Commonly Grouped Processors
 - Are annoying to trace, due to indirection
 - Can reveal more subtle unchecked copies
 - Ex: Is MZ -> Is PE -> Is UPX

Audit Points - Inefficiencies

– Engine vs. Product differences

- Can be an issue when engine is stricter than the product
- Ex: Recent Multi-vendor zip issues

Audit Points - Inefficiencies

– Default Scan Levels

- Can be an issue when product does not require multiple extractions
- Ex: Packed and SFX

Audit Points - Inefficiencies

- File Size Limitations

- Small archives can contain large files

Audit Points - Inefficiencies

– Format Collisions

- Files conforming to multiple formats may be used to trick state and evade detection

O-Day Detection

- Generally very minimal capabilities
 - Measure virus propagation by number of infected customers.
- Evasion?
 - Write a new virus.

Audit Points – Memory Corruption

– Inconsistent Checks

- Length type mismatches can be abused to bypass checks, wrap allocations, and overflow copies
- Negative offsets can be abused to bypass checks and overflow copies

Audit Points – Memory Corruption

– Wrappers

- Allocators that modify length
- Reads that truncate length (reduces chance of access violation on overflow on negative copies)

Audit Points – Memory Corruption

- Error-Prone Formats:
 - 32 bit fields
 - Interesting to examine sign and any calculations
 - Ex: PECOFF – Packed & SFX, Archives

Audit Points – Memory Corruption

- String Based Formats
 - These can be hard to implement correctly
 - StringToNumber conversions are interesting
 - Ex: TNEF, MIME, PDF

Common Error #1

```
MOV ECX, USERINPUTPTR
PUSH ECX
LEA ESI, [EBP-100h]
PUSH OFFSET _ss      ; "%s"
PUSH ESI
CALL _sprintf
ADD ESP, 0Ch
```

Common Error #2

```
MOV EAX, DWORD PTR [EBX]
```

```
CMP EAX, 40h
```

```
JG TOO_LARGE
```

Common Error #3

```
MOV ESI, DWORD PTR [EBX]
LEA EAX, [ESI+18h]
PUSH EAX
CALL malloc
ADD ESP, 4
MOV EDI, EAX
TEST EDI, EDI
JZ ALLOCATION_FAILED

PUSH ESI      // Size to Read
PUSH EDI      // Destination Buffer
PUSH EBP      // File Descriptor
CALL read_file_wrapper
ADD ESP, 0Ch
```

Common Error #4

```
XOR EBX, EBX
```

```
START_LOOP
```

```
MOV AL, [ESI]
```

```
INC ESI
```

```
INC EBX
```

```
TEST AL, AL
```

```
JNZ START_LOOP
```

```
MOVZX ECX, BX
```

```
LEA ESI, [ECX+1]
```

```
PUSH ESI
```

```
CALL malloc
```

```
ADD ESP, 4
```

Another Error

```
MOV EAX, DWORD PTR [ESI]
MOV EBX, DWORD PTR [ESI+4]
ADD ESI, 8
XOR EDX, EDX
DIV EBX
```

Audit Methodology

– Identify Utility Functions

- Naming these will aid in tracing input later
- Ex: Wrappers, FileIO, Allocations

Audit Methodology

– Trace Un-trusted Input

- Examine data that influences:
 - Allocations
 - Copies
- Structure members
 - Initializations are easy to spot
 - Use is less easy – binary search for offset

Audit Methodology

- Reverse File Format Processors
 - Track class member offsets and sizes
 - Will reveal more subtle bugs

Audit Results

– Symantec

- Unchecked offset reconstructing UPX PE header
- Can be triggered by providing a negative offset to prior heap chunk containing MZ header with crafted PE header
- Heap overflow with no character restrictions

Audit Results

– McAfee

- Improperly checked file name and path strlen in LHA level 1 header
- Signature in .dat to detect for malformed LHA file
- Can be triggered by supplying a malformed LHA file, that also conforms to the PECOFF format
- Stack overflow with ascii character restrictions

Audit Results

– TrendMicro

- Improperly checked filename strlen in ARJ header
- Doesn't overflow the next chunk's header, but does corrupt various pointers, which results in the address of the filename being written to an arbitrary destination
- Kernel Heap overflow with ascii character restrictions

Audit Results

– FSecure

- Improperly checked filename strlen in ARJ header
- Standard heap overflow with ascii character restrictions

Future Points of Interest

– Large Files

- Signed Checks
- Type Truncation
- Integer Overflows/Wraps/Underflows
- Ida Examples

Future Points of Interest

– New Formats

- Formats implemented due to bugs
- Formats implemented due to wide use

– Product Administration

Questions?