

Perspectives on Security

Dr. Samuel Mulder

Sandia National Laboratories

September 2013

Program Understanding

- Understanding a program is easiest at the source code level.
- When was the last time you read a good program?

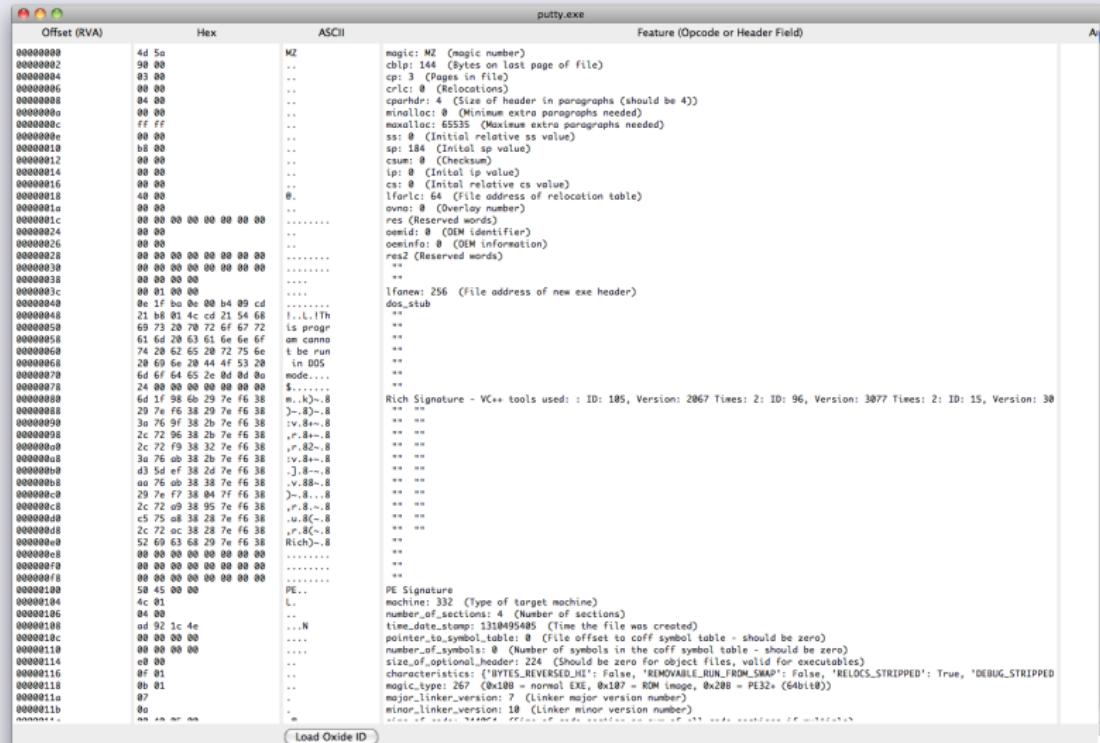
How closely does the program you wrote match what gets run?

- Language transformation
- Compiler optimization
- Libraries and system calls

To really know what gets executed, we must look at the binary image.

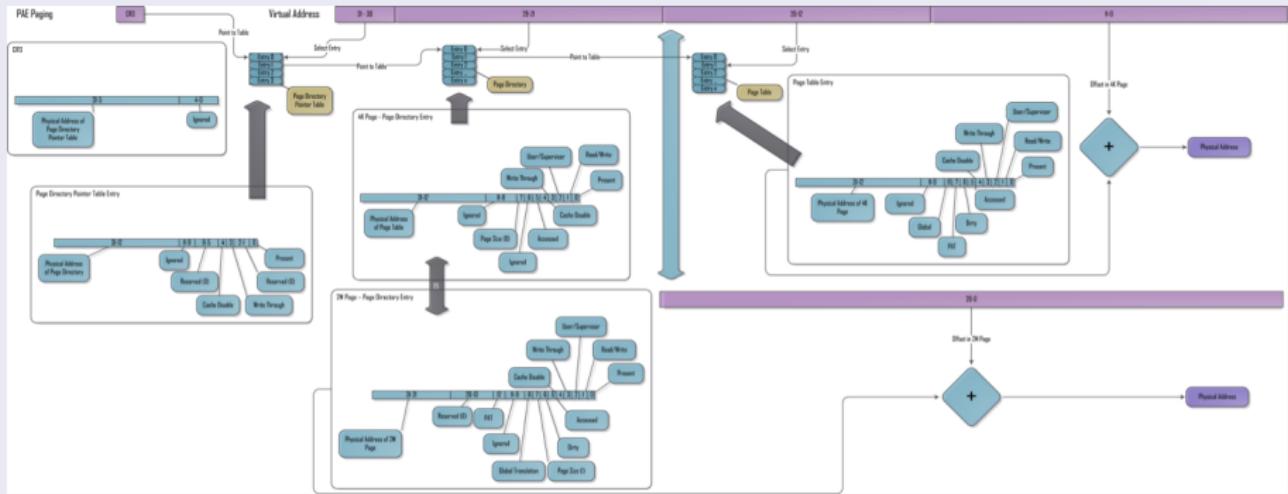
- Disassembly is undecidable.
- Program features like switch statements, inheritance, object-oriented programming all translate into code/data mingling.
- Libraries and system calls bring in new code.

The system loader interprets the PE header.



Offset (RVA)	Hex	ASCII	Feature (Opcode or Header Field)
00000000	4d 5a	MZ	magic: MZ (magic number)
00000002	90 00	..	cbip: 144 (Bytes on last page of file)
00000004	03 00	..	cp1: 3 (Pages in file)
00000006	00 00	..	crlc: 0 (Relocations)
00000008	04 00	..	cpurhdr: 4 (Size of header in paragraphs (should be 4))
0000000a	00 00	..	minloc: 0 (Minimum extra paragraphs needed)
0000000c	ff ff	..	maxloc: 65535 (Maximum extra paragraphs needed)
0000000e	00 00	..	ss: 0 (Initial relative ss value)
00000010	b5 00	..	sp: 184 (Initial sp value)
00000012	00 00	..	csum: 0 (Checksum)
00000014	00 00	..	ip: 0 (Initial ip value)
00000016	00 00	..	cs: 0 (Initial relative cs value)
00000018	00 00	0.	lfarlc: 64 (File address of relocation table)
0000001a	00 00	..	overw: 0 (Overlay number)
0000001c	00 00 00 00 00 00 00 00	res1 (Reserved words)
00000024	00 00	..	osnid: 0 (OEM identifier)
00000026	00 00	..	oeminfo: 0 (OEM information)
00000028	00 00 00 00 00 00 00 00	res2 (Reserved words)
00000030	00 00 00 00 00 00 00 00
00000032	00 00 00 00 00 00 00 00
00000034	00 00 00 00 00 00 00 00
00000036	00 00 00 00 00 00 00 00
00000038	00 00 00 00 00 00 00 00
0000003a	00 00 00 00 00 00 00 00
0000003c	00 00 00 00 00 00 00 00
0000003e	00 00 00 00 00 00 00 00
00000040	00 00 00 00 b4 00 cd
00000042	21 b6 01 4c cd 21 54 68	1..LITH	dos_stub
00000044	69 73 20 70 72 6f 67 72	ls progr	..
00000046	61 6d 20 63 61 6e 6f 6f	om conn	..
00000048	74 62 65 20 72 75 6e	t be run	..
0000004a	29 69 6a 28 44 4f 53 28	in DOS	..
0000004c	6d 6c 6a 20 2e 00 00 00	mode...	..
0000004e	24 60 00 00 00 00 00 00
00000050	6d 1f 98 6b 29 7e f6 38	..,k>..,8	Rich Signature - VC++ tools used: : ID: 105, Version: 2067 Times: 2: ID: 96, Version: 3077 Times: 2: ID: 15, Version: 30
00000052	29 7e f6 38 29 7e f6 38),>..,8	..
00000054	30 7e f6 38 2b 7e f6 38	,v,>..,8	..
00000056	2c 72 96 38 2b 7e f6 38	,r,>..,8	..
00000058	2c 72 f9 38 32 7e f6 38	,r,>..,8	..
0000005a	3a 76 0f 38 2b 7e f6 38	,v,>..,8	..
0000005c	03 76 0f 38 2b 7e f6 38	,j,>..,8	..
0000005e	00 76 0f 38 38 7e f6 38	,v,>..,8	..
00000060	29 7e f7 38 04 7f f6 38	,),>..,8	..
00000062	2c 72 90 38 95 7e f6 38	,r,>..,8	..
00000064	c5 75 08 38 28 7e f6 38	,u,>..,8	..
00000066	2c 72 0c 38 28 7e f6 38	,r,>..,8	..
00000068	52 69 63 68 29 7e f6 38	Rich>..,8	..
0000006a	00 00 00 00 00 00 00 00
0000006c	00 00 00 00 00 00 00 00
0000006e	00 00 00 00 00 00 00 00
00000070	50 45 00 00	PE,..	PE Signature
00000072	4c 01	L,	machine: 332 (Type of target machine)
00000074	04 00	..	number_of_sections: 4 (Number of sections)
00000076	0d 92 1c 4e	..N	time_date_stamp: 1310495405 (Time the file was created)
00000078	00 00 00 00	..	pointer_to_symbol_table: 0 (File offset to coff symbol table - should be zero)
0000007a	00 00 00 00	..	number_of_symbols: 0 (Number of symbols in the coff symbol table - should be zero)
0000007c	c9 00	..	size_of_optional_header: 224 (Should be zero for object files, valid for executables)
0000007e	00 00 00 00	..	characteristics: {'BYTES_REVERSED_HI': False, 'REMOVABLE_RUN_FROM_SWAP': False, 'RELOCS_STRIPPED': True, 'DEBUG_STRIPPED': True}
00000080	0f 01	..	magic_type: 267 (0x10B = normal EXE, 0x107 = ROM image, 0x20B = PE32+ (64bit0))
00000082	0b 01	..	major_linker_version: 7 (Linker major version number)
00000084	07	..	minor_linker_version: 10 (Linker minor version number)

The system loader loads a program into virtual memory.



Programs are like onions...

- The user assumes the application works
- The application assumes the DLLs work
- The DLLs assume the OS works
- The OS assumes the BIOS works
- The BIOS assumes the hardware design works
- The hardware design assumes the physics work

How can we know anything?

- Radical Skepticism?
- Security researchers have to be epistemologists.
- Most of the interesting stuff lies in between layers of abstraction.

Complexity is the enemy of my enemy?

- Which is easier to understand - a Windows executable or a custom embedded system?
- Absolute Knowledge vs. Known Unknowns vs. Unknown Unknowns.

Good vs. Evil

- I just need to find the 'evil bit'.
- Everywhere I look there is badness.

Badness != violating the security properties of the OS

With user level access, a bad guy can:

- Steal all my money
- Steal all my information
- Impersonate me
- Alienate my friends
- Implicate me in criminal activity

With kernel level access, a bad guy can:

- Install a device driver?

Assessing Risk

- Cyber security vs. Physical security.
- Progress is based on accepted risk.

What to do?

- We are accepting more risk on a daily basis, at an alarming rate.
- That will not change.
- We must rise to the challenge.

An effective mindset...

- Effectiveness requires deep humility...
- and amazing hubris.
- Breadth vs. Depth
- A unique adversarial problem space
- A willingness to tackle hard problems