A Highly Immersive Approach to Teaching Reverse Engineering

Golden G. Richard III, Ph.D.
Professor
Director, Greater New Orleans Center for Information Assurance (GNOCIA)
Department of Computer Science
University of New Orleans

Co-founder, Digital Forensics Solutions, LLC

golden@cs.uno.edu
http://www.cs.uno.edu/~golden

What?

- A hands-on course in reverse engineering, focusing on malware
- Provide solid background in theory of reversing
  - Code generation
  - How tools work: e.g., disassemblers, debuggers
  - Anti-analysis and anti-debug strategies
- Interleaved with hard reversing / analysis projects
- Not a collection of Powerpoint and toy examples
- Not a general “hacking” course
  - Not because I object (I don’t)
  - Not enough time in one semester to cover any additional “hacking” topics
- Goal: Students develop serious, usable reverse engineering skills in one semester

Why So Little RE in Academia?

- Because it’s hard for the instructor?
- Perception that skills can’t be developed in a single semester?
- The university won’t allow it
- Should we be doing this?
- Lack of student interest?
- I’m here to discover the others

Aside: Building Trust

- I personally have no problems getting courses like this approved
- I **seriously** lay down the law concerning what will happen to:
  - Classes like this being offered
  - Access to all the cool toys, HW, and SW in my security lab
- …should things go “horribly wrong”
- Historically, despite teaching very hands-on courses in:
  - OS internals
  - Digital forensics
  - Network security
- And despite having classes of students running around with root privileges on the machines in the lab…

Aside: Building Trust (2)

- Nothing external and nothing significant has been destroyed
- Students understand network is monitored and impact of blowing something up outside my lab
- As a result, students are careful and self-policing
- **I've been around for a long time and haven't blown anything up**
- **Your mileage may vary**

Why Do It?

- 60%: RE is useful and **should** be taught
  - Great way to motivate students to dig deeper into systems
  - ASM skills, OS internals, Intel manuals as recreational reading
  - Computing != Computer + Java
- 20%: Students begging
  - Resistance: I knew it would be a lot of work to do correctly, tho it's been coming together for awhile
- 20%: I'm a hacker in professorial clothing
  - Good chance to do what I like

Who?

- Class taught in Spring 2009 for the first time
- 25 students, 2/3 graduate, 1/3 undergrad
- ~20% had taken an OS internals course
- 100% had taken the Intro to Security course
- ~50% had taken or were enrolled in a digital forensics course
- Few had serious assembler skills
- 1 student had nearly expert RE skills
- 2-3 others had at least basic RE skills
- “The hardest course I’ve ever had”
- 1 student dropped in Spring 2009

Aside: ASM Courses: Don’t Get Me Started

- Serious problem: Students have poor ASM skills
- Don’t know about yours, but our ASM course is (IMO) worthless
- Didn’t use to be…I took that course in 1983!
- Can’t volunteer to teach that course…no time
- No time to “teach” the ASM course inside RE
- Solution:
  - (Nearly) compassion-free immersion
  - ASM every day
  - Tight deadlines assignments requiring ASM comprehension

Topics

- Goals of reverse engineering
  - Software interoperability, patch verification, **malware analysis,** **cracking**
- Ethics and legal issues
  - DMCA, EULAs, RE == jail, seek ye lawyers
- Techniques / Tools for RE
  - Static vs. dynamic analysis, disassemblers, debuggers, live forensics tools, memory dumpers, packing / unpacking, …
- Malware background
  - Types, propagation strategies, payload delivery, poly- and metamorphic malware, …
- Basic Intel assembler (a few lectures, then "on the job")
  - Registers, flags, common instructions, data formats, 32 vs. 64bit code, hardware components, paging, debugging architecture, examples

Topics (2)

- Windows Portable Executable (PE) format
- C control structure, function, array, struct/union patterns generated by common compilers
- Common malware functionality
  - Delta offset calculation, API address discovery, infection and propagation, …
- Anti-debugging / anti-VM functionality
  - Dynamic jumps, instruction prefetch attacks, LDT/GDT/IDT location analysis, use of debugging facilities
- Packing and unpacking techniques
  - Hand-rolled, UPX, Armadillo, …

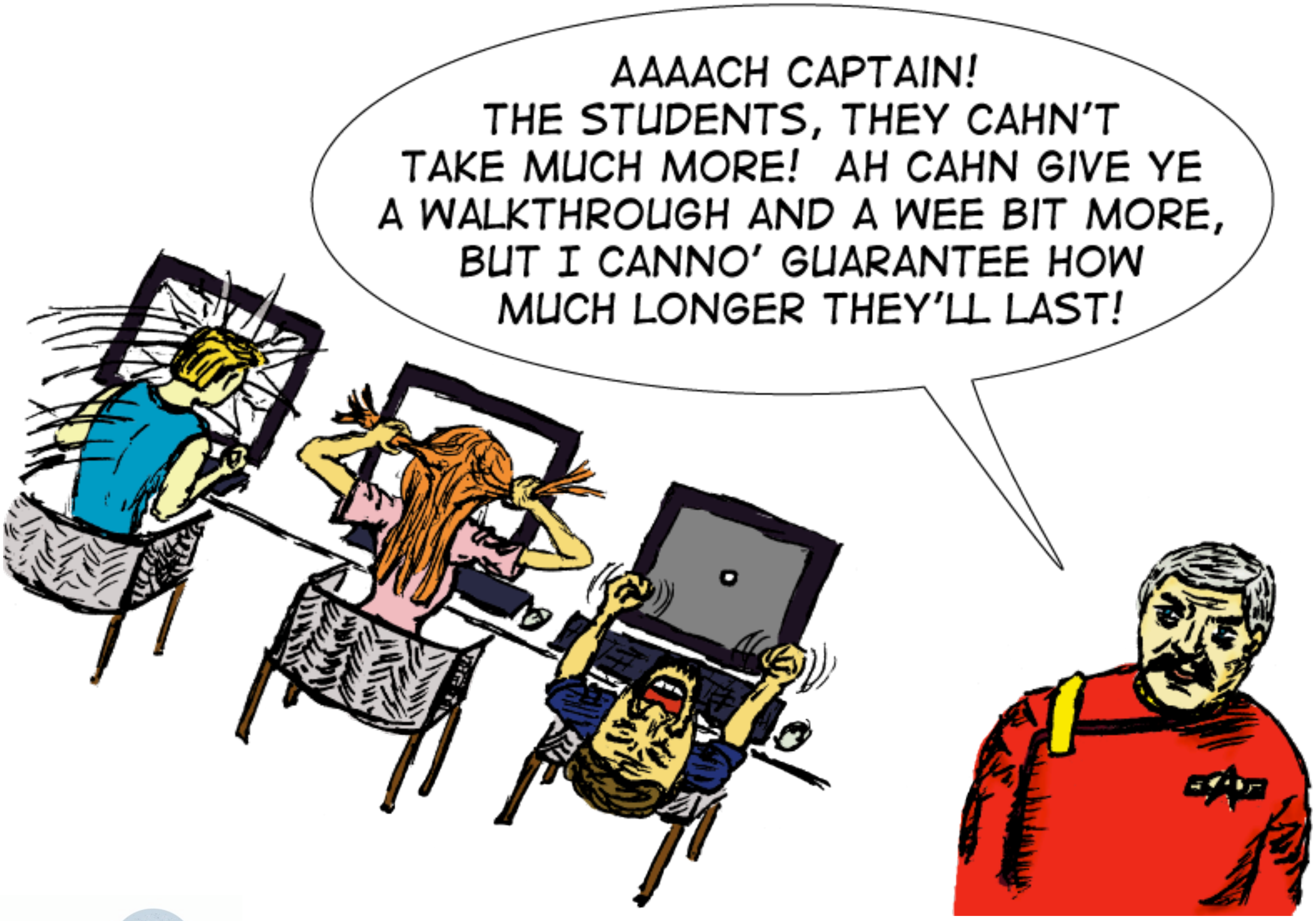
Laboratory Setup

- Isolated gigabit network with fast, private fileserver (16 x 15K SAS drives) – has to serve VMWare images
- Workstations running Linux + VMWare
- User accounts including XP VMWare image stored by file server
- XP image contains:
  - sysinternals suite
  - Visual C++ Express Edition
  - MASM32
  - ollydbg
  - IDA Pro 5.x + x86emu plugin for x86 emulation
  - HBGary Responder (thanks, Penny!)
  - **FACE**, Volatools, ptfinder, …
- Networking OFF in VMWare image whenever possible

Approach: Challenges

- Time is short!
- ASM skills
- Flipping Powerpoint guaranteed to fail
- Want actual, rather than theoretical, skills to emerge
- Skills at end of semester should be (almost?) sufficient to analyze modern malware
- Must **hurt** students (a lot) to achieve skill levels without completely discouraging them

WHEN SCOTTY TEACHES REVERSE ENGINEERING:


AAAACH CAPTAIN! THE STUDENTS, THEY CAHN'T TAKE MUCH MORE! AH CAHN GIVE YE A WALKTHROUGH AND A WEE BIT MORE, BUT I CANNO' GUARANTEE HOW MUCH LONGER THEY'LL LAST!

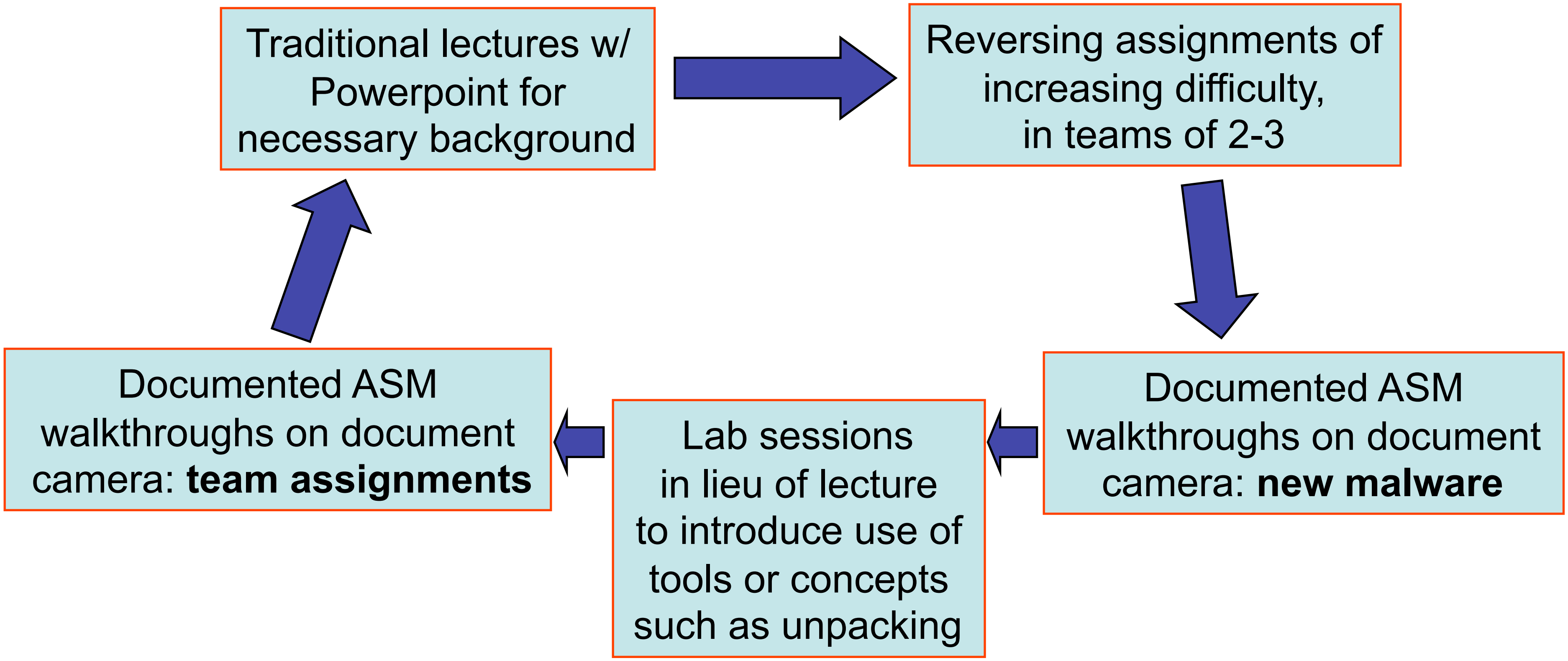
Digital Forensics Solutions LLC
Digital Investigation and Data Recovery

THE UNIVERSITY of NEW ORLEANS

Approach: Malware Sampler

- Requirements:
  - Students start RE immediately
  - With each new malware sample, push students **almost to breaking point** ☺ but not quite
- Michelangelo → DOS-7 → SQL Slammer → Murkry → Lucius → Harulf → Conficker
- These were interleaved with short "malware" samples (that I wrote) to introduce:
  - Registry hacking
  - Replacement of system binaries
  - Addition of user accounts
  - …

Approach: Workflow



Traditional lectures w/ Powerpoint for necessary background

Reversing assignments of increasing difficulty, in teams of 2-3

Documented ASM walkthroughs on document camera: **new malware**

Lab sessions in lieu of lecture to introduce use of tools or concepts such as unpacking

Documented ASM walkthroughs on document camera: **team assignments**

Midterm / Final:
60% reverse engineering assignments
40% background material

Approach: Assignments

- Series of team-based malware analyses
- Goal is to produce fully documented disassemblies
- Initially, uncommented but correct disassemblies
- Later, only a binary malware sample
  - Must coax tools to generate correct disassembly
  - Deal with packing, anti-analysis techniques
- Modest expectations initially, increase sharply as the semester progresses
- In some cases:
  - Solutions accepted and signed
  - Necessary concepts for complete solution discussed in class
  - Solution returned and then may be resubmitted
- Always let students try (and potentially fail) before giving away the solution

```
NukeHD:
        sub cx,cx

NukeDism:
        inc cx
        push cs
        pop es
        mov ax,FE05h
        jmp $-2

        sub ax,E702h
        mov bh,1
        mov dx,80h
        int 13h
        jmp short NukeDism
```

```
NukeHD:
        sub cx,cx            ; cx == sector number <-- 0
                             ; FALL THROUGH...
NukeDism:
        inc cx               ; target next sector
        push cs              ;
        pop es               ; es <-- cs
        mov ax,FE05h         ; ax <-- FE05h
        jmp $-2              ; jumps into middle of last instruction
                             ; last instruction disassembled =
                             ; B8 05 FE EB FC
                             ;
                             ; JMP targets 05 byte which is the
                             ; opcode for a 16-bit immediate add
                             ; to AX, thus ax <-- ax + EBFEh
                             ;
                             ; the remaining byte, FC, is the
                             ; opcode for the single byte instruction
                             ; CLD (clear direction flag)
                             ;
        sub ax,E702h         ; ax <-- ax - 0E702h = 301h
        mov bh,1             ;
        mov dx,80h           ; first hard drive
        int 13h              ; write 1 sector to hard drive
        jmp short NukeDism          ; write "forever"
```

Approach: Exams

- **30%: Abstract scenarios / “Book material”**
  - “You discover that a binary is packed with UPX. To discover the original entry point (OEP), you…”
  - “A malware sample makes heavy use of dynamic JMPs. Which disassembler design is more likely to encounter problems? Why? Solutions?”
- **70%: References to RE exercises**
  - Precise, detailed answers required
  - Hard to answer within available time if student didn’t participate in the team-based analyses
  - “When you analyzed the following section of Harulf, what did you discover? Comment each line.”
  - Example follows on next slide

```
Start:
        jmp stuck
        sig_1 dd 0
        sig_2 dd 0
stuck:
        call here

        jmp getdelta
here:
        assume fs:nothing
        mov eax,[esp]
        push eax
        push fs:[0]
        mov fs:[0],esp
        xor eax,eax
        mov eax,[eax]
        ret
getdelta:
        ...
        pop fs:[0]
        pop edx
        pop ebp
        sub ebp,offset here
        add ebp,2h
        cmp ebp,0
        je skipdecrypt
```

```
Start:
        jmp stuck
        sig_1 dd 0
        sig_2 dd 0
stuck:
        call here         ; start delta offset calculation,
                                    ; trip up debuggers with stack-based SEH
        jmp getdelta                ; this will be new SEH
here:
        assume fs:nothing
        mov eax,[esp]               ; address of “jmp getdelta” in eax
        push eax                    ; save address on stack (new SEH)
        push fs:[0]                 ; save old SEH head
        mov fs:[0],esp              ; “jmp getdelta” is new SEH
        xor eax,eax                 ; zero eax
        mov eax,[eax]               ; null ptr reference, invokes SEH
        ret
getdelta:
        ...
        pop fs:[0]                  ; restore SEH
        pop edx                     ;
        pop ebp                     ; address of getdelta
        sub ebp,offset here         ; subtract compile-time offset of ‘here’
        add ebp,2h                  ; jmp getdelta is two bytes
        cmp ebp,0                   ; are we at entry point?
        je skipdecrypt     ; yes, no need to decrypt body
```

Final Thoughts

- It’s fun
- It’s hard (for you and for students)
- Lots of initial student interest, interest sustained
- Student feedback was overwhelmingly positive
- Great way to generate students with sufficient background in systems to do real research
- Potential benefit to students is high
- In many cases, job interviews are “won” with a single data point—this course provides many
- RE will be offered regularly at UNO

Thanks.

?

golden@cs.uno.edu

golden@digitalforensicssolutions.com