The Importance of Intensive Digital Forensics Education

Golden G. Richard III

Professor of Computer Science and University Research Professor
Director, Greater New Orleans Center for Information Assurance (GNOCIA)
University of New Orleans

GIAC-certified Digital Forensics Investigator
Founder, Arcane Alloy, LLC

golden@cs.uno.edu
http://www.cs.uno.edu/~golden
Who is This Guy? / Disclaimers

• University Research Professor and Professor of Computer Science @ University of New Orleans
• Director, Greater New Orleans Center for Information Assurance (GNOCIA)
• GIAC-certified Digital Forensics Investigator
• Member of the United States Secret Service Cybercrime Taskforce
• Member of the American Academy of Forensic Sciences (AAFS)
• Founder, Arcane Alloy, LLC

(c) 2013 by Golden G. Richard III / golden@cs.uno.edu
Concert Photographer by Night

(c) 2013 by Golden G. Humphrey
Definition: “Tools and techniques to recover, preserve, and examine digital evidence on or transmitted by digital devices.”

Computers, PDAs, cellular phones, videogame consoles, digital cameras, copy machines, printers, digital voice recorders...
What That Really Means

- Data. “You only think it’s gone.”
- Sensitive data tenaciously clings to life.

- The vast majority of users—and some digital forensics investigators--have no idea what’s really stored on their digital devices...
- ...and no ability to properly “clean up” even if they do suspect what’s there

(c) 2013 by Golden G. Richard III / golden@cs.uno.edu
Unfortunately

• Misinformation Abounds
• About what can and can’t be recovered from digital devices
• And how hard (or not) data might be to recover
• About how computer systems and applications work!
• Many digital forensics “experts”…aren’t.
Why Should Anyone Care?

• Privacy is good
  – In many cases, it’s the law
  – Knowing what’s stored and how to control access and securely destroy data is important
  – 99% of users only think they know

• Implications for policy makers
  – Cannot simply tell employees “protect data”
  – They only think they’re doing it right
  – Education! Training!
  – One slip and the data’s out!

• When private data leaks
  – Financial losses, embarrassment, heads rolling
  – Very ugly
  – Digital investigation increasingly important in civil and criminal litigation

(c) 2013 by Golden G. Richard III / golden@cs.uno.edu
Digital Forensics Investigation

• What’s possible?
  – Recovery of deleted data
  – Discovery of when files were modified, created, deleted, organized
  – Can determine which storage devices were attached to a specific computer
  – Which applications were installed, even if they were uninstalled by the user
  – Which web sites a user visited
  – Recovery even when drives / media are in “broken”
  – Live forensics:
    • Running processes (+ terminated ones)
    • Live network connections (+ dead ones)
    • Extraction of important kernel and application data
    • Malware extraction
Examples of Digital Evidence

- Documents
- Threatening emails
- Suicide notes
- Bomb-making diagrams
- Malware
- Child pornography (contraband images/videos)
- Evidence that network connections were made between machines
- Cell phone SMS messages
- Deleted voice messages on digital voice recorder
- Deleted copy jobs on laser printer
- Anything that can be stored on digital devices

(c) 2013 by Golden G. Richard III / golden@cs.uno.edu
Where’s the Evidence?

- Undeleted files + filesystem metadata
- Deleted files
- Application metadata
- Windows registry
  - e.g., USB device histories, recently accessed files, typed URLs, etc.
- Print spool files
- Hibernation files
- Temp files (all those .TMP files!)
- Log files
- Slack space
- Swap files
- Browser caches
- Network traces
- RAM
Live Forensics: Evidence

- Running processes
  - open DLLs
  - registry
  - file handles
- Open files
- Network connections
- Raw Memory
- Regular disk files
- Images of entire disk
  - Live disk imaging
- Deleted files
  - Live file carving

- e.g., discover unauthorized programs
- e.g., detect keystroke loggers
- e.g., find "shreds" of volatile evidence, analyze kernel structures

Do “traditional” forensics on a live machine

(c) 2013 by Golden G. Richard III / golden@cs.uno.edu
Live Forensics: Evidence (2)

• Volatile registry contents
  – Some portions of the Windows registry are volatile—not mirrored in on-disk hives

• Kernel structures
  – e.g., Linux slab allocator: can reveal lots of current and historical information
    • Recent NAT addresses
    • Network packet data
    • ...

• Live forensics is where incident response / malware analysis / reverse engineering crosses paths with digital forensics

• Arguably, all of these areas are essentially becoming a single one, whatever we choose to call it ("DFIR")

• More on this soon
Digital Forensics Process

• Legal: Balance need to investigate vs. privacy rights
• Identification of potential digital evidence
  – Where might the evidence be?
  – Which devices did the suspect use?
• Preservation and copying of evidence
  – On the scene...
  – First, stabilize evidence...prevent loss and contamination
  – If possible, make identical copies of evidence for examination
  – Copies can be made on the spot, or more usually, in the lab
• Careful examination of evidence
  – File recovery / File carving
  – Keyword searches
  – Generation of timelines
  – Examination of the registry
  – ...
• Presentation of results (deposition, court testimony, etc.)
On the Scene Preservation

“Dear Susan,
It’s not your fault…

Just pull the plug?
Move the mouse for a quick peek?
Do a live analysis?

Volatile computing

Living room

Tripwires

wireless connection

Basement/closet

(c) 2013 by Golden G. Richard III / golden@cs.uno.edu
Careful Documentation is Crucial
Why Does Digital Forensics Work?

• (Benevolently?) prey on mechanisms designed for **performance, not privacy**
• Creative uses of data intended mostly for other things
• Correlation of multiple data sources to create richer context
• In some cases: logs, etc. actually meant to be used for forensic purposes, but not the general case
Visualization of 256MB USB Thumb Drive
→ FAT32 format
→ NTFS format
→ ext3 format
Before Format
After Format
Dear Sir,

To whom it may concern,

I'm writing to apologize for my rude behavior on the night of August 15, 2002, when I attended your party and ate every single piece of tuna sushi. Your daughter is preciously cute.

I can't freaking believe I'm writing this crap.

I'm writing to apologize for my rude behavior on the night of August 15, 2002, when I attended your party and ate every single piece of tuna sushi. Your daughter is preciously cute.

I hope we can put this episode behind us. Thanks.

--Golden
From: "sherwan" <XXXXXXX10@hotmail.com>
Subject: tolstoy
Date: Fri, 27 Feb 2004 02:44:59 -0500
MIME-Version: 1.0
Content-Type: text/plain;
    charset="iso-8859-1"
Content-Transfer-Encoding: 7bit
X-Priority: 3
X-MSMail-Priority: Normal
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2800.1165

ignorant materialism of Ilyich's peers by pointing out their concern with professional advancement and social status

jane
Normal
jane.ullah
Microsoft Word 9.0
Lincoln University

(c) 2013 by Golden G. Richard III / golden@cs.uno.edu
Browsing History

Internet Explorer Browser History
That Web Activity...

- Internet Explorer browser cache loaded with thousands of “NSFW” images
- Appears that the user spent her entire day doing nothing but downloading porn
- Images are located in precisely the right places to indicate web browsing activity
- Times that images are downloaded correspond to times the user was “working” at the computer
- The problem?
- The user didn’t actually download any of the images
- She wasn’t even using IE when the images were downloaded!
Aside: The Trojan Defense

• Trojan defense is popular in, e.g., child porn cases
• “I didn’t download that stuff—a virus must have done it”
• Generally, not taken very seriously by investigators
• Run antivirus, find nothing, assume user was lying
• Unsophisticated investigators can’t really do much more, anyway
This Time, It’s Real

• Extremely difficult to detect malware is the culprit in case (from actual, recent casework)
• Not explicitly detected by any commercial antivirus
• Also extremely difficult to analyze
  – Packed, crazy control flow, anti-debugging tricks
• Mimics IE browsing to download porn using the IE API, in the background!
• Only downloads porn when there’s computer activity!
Deeper Skills, Not Just Button Pushing!

• Huge demand for students with strong cyber skills that can crack cases like that one
• Digital forensics skills should include ability to:
  – Do live forensics
  – Do reverse engineering
  – Deeply understand internals
  – Deeply understand offensive techniques
  – Really take things apart when it’s necessary
• Churning out more GUI-enslaved, button-pushing investigators is NOT a complete solution!
• (Although we probably need some)
• Plus, we need people to develop better tools

(c) 2013 by Golden G. Richard III / golden@cs.uno.edu
I once thought this was obvious, but:

Digital forensics investigators, reverse engineers, malware analysts, pen testers can’t simply be “computer users”

• Relying too much on abstractions to shield complexity is:

  Advantage: Attacker

• Attackers don’t play by an abstraction’s rules
• Attackers read the specs and break them
• Attackers read the code and understand it—deeply
• Our students need to be able to do this, too
WTF?

• Yet I still hear, even in cyber exercises (!):
  • “Don’t teach the students to hack.”
  • “Don’t teach them offensive techniques.”
  • “Just show them what software to run to protect things.”
  • “Just show them which buttons to press to find the evidence.”
  • “Just show them how to defend.”
  • “They don’t need to know how to do that stuff.”
• Sorry, but I think I will (and we should) teach them “that stuff”
More Motivation for Deep Skills: An eBlaster Case

Computer infected with eBlaster, (commercial malware) from Spectorsoft

Extremely stealthy! How do you even know it’s there?

See credit card statement of someone living in your household!

Try default hotkey sequence: SHIFT-ALT-CONTROL-T

Uses code injection, hooks system calls, and other hiding techniques to make it difficult to detect

(c) 2013 by Golden G. Richard III / golden@cs.uno.edu
Hotkey sequence brings up authentication screen.

Can’t access config screen without the password.

Process is spawned to handle this dialog (dies after dialog is dismissed!)
dd image → raw2vmdk → VMWare virtual machine → Live forensics analysis
DumpIt window
(dumps physical memory)
The software is stealthy and fairly difficult to analyze. The hope is that developers “blew it” and leaked the plaintext password in memory somewhere…

Enter sample password “VERYUNIQUE” in password dialog.

Don’t dismiss the “invalid password” dialog.

Dump memory in the VM while the dialog box is displayed.

YOUR-US67PI6LUV-20121017-214253.raw
# gstrings --o --e | YOUR-US67PI6LUV-20121017-214253.raw | grep VERYUNIQUE > strings.txt

467105092:VERYUNIQUE

# python vol.py --profile=WinXPSP3x86 -f YOUR-US67PI6LUV-20121017-214253.raw strings -s strings.txt --output-file=stringslocation.txt -S

1bd77544 [3772:0012f544] VERYUNIQUE

PID of process whose address space includes string

# python vol.py --profile=WinXPSP3x86 -f YOUR-US67PI6LUV-20121017-214253.raw pslist

... 0x82c24bf0 xmlavipv.exe 3772 1592 2 78 0 0 2012-10-17 21:41:20

# python vol.py --profile=WinXPSP3x86 -f YOUR-US67PI6LUV-20121017-214253.raw -p 3772 memdump
Activity Reports, Forwarding Services and Alerts are sent via email using the following settings:

**Report Delivery Summary**

Based on your current settings, eBlaster will send an Activity Report via email every 60 minutes to 'brgolden@unomaha.edu'.

eBlaster will automatically forward Emails (including attachments), Chat/Instant Messages and Keyword Alerts to 'brgolden@unomaha.edu'.

**Activity Reports**

- **Activity Report Delivery**: On/Off
- **Send a Report**: Every 60 minutes
- **Format Report as**: HTML/Plain Text

**Email**

- **Forward All Emails**: On/Off
- **Include Attachments**: On/Off

**Chat / Instant Message**

- **Forward All Chat/IMs**: On/Off

**Alerts**

- **Forward Keyword Alerts**: On/Off
A Reverse Engineering Course, Too

- **My approach:**
- A hands-on course in reverse engineering, focusing on malware analysis
- 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
- Goal: Students develop serious, usable reverse engineering skills in one semester
- Proven: With many students getting $100-150K jobs right out of my program

(c) 2013 by Golden G. Richard III / golden@cs.uno.edu
Very Little RE in Academia

• Because it’s hard for the instructor?
  – Yes. Absolutely.

• Skills can’t be developed in a single semester?
  – In my experience, people believe this.
  – But they’re wrong. 😊

• The university won’t allow it.
  – I didn’t ask first. 😊
  – But there are ways to make it work (ask me later).

• Should we be doing this?
  – Yes.

• Lack of student interest?
  – Absolutely not—class fills every time and zero drop rate.
Why Did I 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, though it’s been coming together for awhile

• 20%: I’m a hacker in professorial clothing
  – Good chance to do what I like
Aside: Assembler

• Serious problem: Students have poor ASM skills
• No time to “teach” an ASM course inside RE
• Instead, on the job
• 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, ...

• Intensive Intel assembler “on the job”
Topics (2)

• Executable file formats (e.g., PE)
• 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 derivatives, Armadillo, Themida, etc.
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!
Approach: Workflow

- Traditional lectures w/ Powerpoint for necessary background
- Lab sessions in lieu of lecture to introduce use of tools or concepts such as unpacking
- Reversing assignments of increasing difficulty, in teams of 2-3
- Documented ASM walkthroughs on document camera: **team assignments**
- Documented ASM walkthroughs on document camera: **new malware**

(c) 2013 by Golden G. Richard III / golden@cs.uno.edu
Approach: Assignments

• Series of team-based malware analyses
• Initial goals are to produce fully documented disassemblies
• Initially, provided 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"
And Offensive Computing

• “Deep” exploits that target multiple OS components are “magical” to abstraction-mongers
• Essential that investigators can deal with, e.g., very sophisticated rootkits
• My solution:
• 2nd OS course (our “internals” course) is now an offensive computing class
• Focuses on teaching operating systems internals in the context of analysis of real, complex OS kernel attacks
• There’s nothing like actually “touching” real exploits
• Motivating example from that class: one min version
set_selection() Memory Corruption

• Console subsystem in Linux allows text selection (by GPM mouse daemon)
• Bug is in translation of characters represented as 16-bit integers into 3 byte UTF-8 Unicode during selection with mouse
• Buffer can be up to 2 characters too short
• With care, allows 1-2 byte memory overflow in Linux kernel

(c) 2013 by Golden G. Richard III / golden@cs.uno.edu
/* Allocate a new buffer before freeing the old one ... */
/* chars can take up to 3 bytes */
multiplier = use_unicode ? 3 : 1;
bp = kmalloc((sel_end-sel_start)/2*multiplier+1, GFP_KERNEL);  [3] 
[...]
/* Fill the buffer with new data ... */
for (i = sel_start; i <= sel_end; i += 2) {                     [4]
c = sel_pos(i);
if (use_unicode)
    bp += store_utf8(c, bp);                                    [5]
else
    *bp++ = c;

Oops…sel_end – sel_start + 1
Buffer is 2 bytes too short!

Main idea: For every character in selection, convert it to UTF-8 if Unicode is in use in the kernel, otherwise just store it directly. Characters in the selection in the console are 16-bit integers that represent an index into the current font; internal representation is 3-byte UTF-8 Unicode.
struct file_operations {
    struct module *owner;
    loff_t (*llseek) (struct file *, loff_t, int);
    ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
    ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
    ssize_t (*aio_read) (struct kiocb *, const struct iovec *, unsigned long, loff_t);
    ...
    int (*readdir) (struct file *, void *, filldir_t);
    unsigned int (*poll) (struct file *, struct poll_table_struct *);
    int (*ioctl) (struct inode *, struct file *, unsigned int, unsigned long);
    int (*open) (struct inode *, struct file *);
    int (*flush) (struct file *, fl_owner_t id);
    ...
};

static struct file_operations timer_list_fops = {
    .open = timer_list_open,
    .read = seq_read,
    .llseek = seq_lseek,
    .release = single_release,
};

struct sctp_stream {
    __u16 *ssn;
    unsigned int len;
};

struct sctp_ssnmap {
    struct sctp_stream in;
    struct sctp_stream out;
    int malloced;
};

(c) 2013 by Golden G. Richard III / golden@cs.uno.edu
After overflow at 473:

\[ 22 \times 4 + 4 + 10 \times 2 = 68 \]

Free list

Overlapping misaligned object to be

Set selection buffer

Overflowed next pointer changes from 512 (8x200) to 700 (8x280)
After 474-475 (two more objects)

\[ \frac{2}{3} \text{ allocated} \]
After 476-477 (last two objects)
Conclusion

- Digital forensics education is essential, for lots of reasons
- “High tech” skills are even more essential
- Easy stuff: clicking buttons in a forensic suite, like FTK or Encase
- Harder stuff requires:
  - Incident response skills
  - Live forensics analysis
  - Reverse engineering
  - Knowledge of offensive capabilities
- Without those skills, in many cases, you get nothing
- Or worse, the wrong “answers”
- Huge opportunities for students who can do it right!
- Tremendous challenges and fun for faculty who want to do it right!

Discussion?