The Impact of Secure Software on Homeland Security

Ron Moritz, CISSP

Senior Vice President, Chief Security Strategist

Computer Associates

eTrust™ Security Solutions

Executive Summary

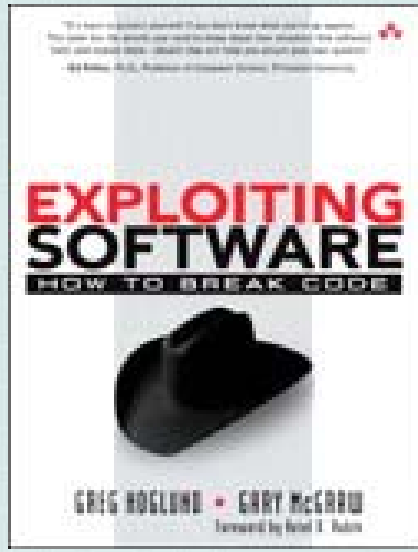
- Abstract (Executive Summary):
  - More and more we hear the call for delivering secure software. Secure software is different from security software. In fact, security software is often the band-aid response to our inability to write secure code. Fixing the way we design and manufacture software products is a key strategy in the nation's war on cyber-crime and -terrorism. Is this goal illusive? Is our only response to cyber-threat reactive? Or are we prepared to confront this challenge by changing our approach to software development? It is irresponsible for software engineers, whether employed by government or private industry, whether writing code for a software publisher or for proprietary use, to neglect their obligation to society to deliver safe programs. Arguably, it is their patriotic duty to ensure that software includes safety as a primary design objective. In this session, we will discuss the need to overhaul current software engineering thinking with respect to software quality and security.
- Learning Objectives:
  - Why is today's software engineering methodology is flawed?
  - Why are software flaws a key challenge to cybersecurity?
  - Why should we modify the way we write software immediately?
  - How do we measure software quality and secure software?
  - What incentives must we offer to software developers?

Topics for Discussion

- Software is the problem
- Band-aids are not the answer
  - (or “patching is icky!”)
- The way we think is flawed
  - (still reactive after all these years)
- Safety as a primary design objective
- Changing the way we design and manufacture

Credits

- Greg Hoglund and Gary McGraw
  - “Exploiting Software: How to Break Code”
- Michael Howard and David LeBlanc, Microsoft
  - “Writing Secure Code”
- DHS Security Across the SDLC Task Force
  - http://www.cyberpartnership.org/init-software.html



Software Security Defined

- Finding implementation errors
  - Can scan code for implementation level mistakes
  - Cannot code human experience in detection
- Securing applications
  - Can ensure that only approved software is executed
  - Cannot find vulnerabilities in executables (changing)
- Security solutions
  - Can install firewalls, anti-virus, crypto
  - Cannot contain exploits due to code vulnerabilities

It's Not the Network Traffic, Stupid …

- … It's the software
  - that processes the packets that are allowed through
- There are no "silver-bullet" point solutions
  - "Security is not a product … it's a process"
  - "Security is not a destination … it's a journey"

I'd rather buy SECURE software than SECURITY software

Break the Software, Own the World

- Software issues include
  - Malformed killer packets
  - Buffer overflows
  - Application defects
- The root cause is software failure
  - Faulty inputs result in crash
  - Faulty controls allow injection and execution of code
  - Faulty oversight allows privileged command execution

Who Loses When Software Fails?

- Business
  - Supply chains, manufacturing plant controls, customer relationship management systems, world-wide corporate communications
- Consumers and Partners
  - Confidential data exposed to unauthorized people
- Everyone and Everything
  - Inability to share data, programs, and computational resources
  - There goes the on-demand neighborhood and web services!

Why We Are Where We Are?

- Go-go 90s and market pressure
  - "Where's the new-new thing, chief?"
  - "Needed it yesterday"
  - "Time-to-Market"
- Time-to-Money
  - If it doesn't go out the door then there is no money in!
- Doing it right is too expensive (or is it?)
  - Result: poorly written / tested code
  - Result: many exploitable bugs
- Truth?
  - We all slept through our software engineering class!
  - Experts in secure software development are few / far between

Software and the Critical Infrastructure

Cyber Crime, Cyber War, Cyber Terrorism

Conflict in the information age, where cease fire and peace are unknown and information superiority is more critical than air superiority, lives on a battlefield that is no longer geographically confined.

What does the fact that information superiority means dominance mean to government and business leaders?

Conflict in the Cyber Battle Space

- War is no longer geographically confined
  - n-dimensional cyber battle space
- Every tech ops center deals with digital warfare
  - Every individual computer and network user is on the front line
- Malicious actors with political, financial and entertainment-related motivations attack:
  - home users (become zombies and sleepers),
  - corporate enterprises,
  - governments, and
  - critical infrastructure providing essential services

Why Do We Go To War?

- Three reasons:
  - Women
  - Economy (land, resources)
  - Religion
- We're at war today (not the visible kind)
  - East versus west
  - Information is the core
  - Economics of knowledge workers (off-shoring)
- When battle space is n-dimensions, does the United States remain a super-power?

When the Battle Space is N-Dimensions …

- … does the United States remain a super-power?
- Nations with InfoWar capability at an advantage
  - Key: Gather and control large amounts of info
- IW tied to intelligence gathering (espionage)
  - National defense and security
  - Assistance in military operations
  - Expansion of political sphere of influence
  - Increase in economic power / market share
- Risk = Exposure
  - Hide in the “cloud”
  - Cost of intelligence gathering is way down

Software and the Critical Infrastructure

- Today's bank is not gold but bits
  - No gold bullion in the vault
    - "cloud of electrons at the right place at the right time"
  - Money is represented electronically
    - Trillions of e-$ flow through nations daily
- Controlling the global networks means controlling the global economy (IW goal)
  - Today's experiments may be tomorrow's domination
    - Testing the limits, boundaries, capability, cyber-weapons
    - Non-hobbyists testing our ability to respond

What does this mean to business leaders?

CEOs who do not understand or recognize the importance of their information systems and do not invest in the security of these systems will fail.

Digital Tradecraft

- Tradecraft:
  - The technical skills used in espionage including lock picking, secret writing, clandestine photography, surveillance, dead drops
- Digital tradecraft:
  - Leverage software weaknesses and vulnerabilities in order to gain access to information or services or resources

Digital Tradecraft

- Exploits Software Weaknesses
  - Configuration errors
  - Software bugs
  - Design flaws
- Introduces Subversive Code (can do anything!)
  - Logic bombs
  - Spyware
  - Trojan Horse

Subversive Code

- Data Collection
  - Packet sniffing
  - Keystroke monitoring
  - Database siphoning
- Stealth
  - Hiding data (stashing log files)
  - Hiding processes
  - Hiding users of a system
  - Hiding a digital “dead drop” (aka, music server)

Subversive Code

- Covert Communication
  - Allowing remote access without detection
  - Transferring sensitive data out of the system
  - Covert channels and steganography
- Command and Control
  - Allowing remote control of a software system
  - Sabotage (variation of command and control)
  - Denying system control (DoS)

Digital Tradecraft End Goal

- Exploit software in order to construct and introduce subversive code
- The software attacker can be really dangerous
  - “Digital Pearl Harbor” is not a fiction
- Real because software can fail spontaneously
  - Even without mal intent software is fragile
  - Add an adversary intent on making your system break
- Bad software is a real problem
  - Exploiting bad software is the concern

Bad Software Resulting in Failure

- NASA Mars Lander Crash
  - Translation between English and metric units
  - Cost: $165M
- Denver Airport Automated Baggage System Failure
  - Carts unable to detect or recover from failure
  - Delayed airport opening by 11 months @ $1M/day
- MV-22 Osprey Advanced Military Aircraft Crash
  - Faulty hydraulic line burst on take off
  - No backup system engaged due to software failure
  - Cost: 4 Marines killed
- U.S. Vicennes Missile Launch
  - “Cryptic and misleading output” from radar tracking system
  - Identifies hostile threat, a fighter jet, and launches missile
  - Cost: 290 people on a commercial Airbus 320

Functionality Has Trumped Security

- "I Love You" virus and Microsoft Outlook
  - Outlook designed to execute programs received
  - Virus exploited built-in scripting feature of Outlook
  - Possible even after years of Office macro exploits

Software Risk Management Dependencies

- Complexity: Bugs per KLOC
  - 5 B/KLOC in QA-tested software
    - QA testing includes fault injection and failure analysis
  - 50 B/KLOC in feature-tested (commercial) software
- Examples
  - Solaris 7 = 400 KLOC → 2K – 20K bugs
  - Linux = < 1.5 MLOC → 7.5K – 75K bugs
  - Windows NT = 35 MLOC → 175K – 1.75M bugs
  - Windows XP = 40 MLOC → 200K – 2.00M bugs
  - Boeing 777 = 7 MLOC → 35K – 350K bugs

Bugs On Top Of Bugs (on top of bugs …)

(More Questions Than Answers)

- How many problems will results in security issues?
- How are bugs / weaknesses turned into exploits?
- What happens when complex applications are placed on top of complex operating systems?
- Can this even be avoided when underlying coding languages (C/C++) fail to help overcome even well understood (and relatively simple) coding errors like buffer overflows?
- And what about malicious code intentionally injected and hidden inside complex code?
  - Is that NSA code hiding inside that flight simulator?

Leveraging Software Extensibility

- Worm v. Virus
  - Virus spreads through infected executables
  - Worm spreads over networks without depending on infected executable
- Attack Worm/Virus: Mobile code based weapons
  - Propagate
  - Install back doors
  - Monitor systems
  - Compromise machines for later use (e.g., zombie)
  - Exploit a vulnerability as widely as possible
- New phones with embedded OS and filesystem
  - What happens when virus infects the cell phone network?
  - SPAM on the VoIP channel (SPIT)?

The Math Favors the Attacker

- Success of worms like Blaster are logical
  - Consider 30K node network (mid-size corporation)
  - Each node has an average of 3,000 exe modules
  - Each module is about 100KB
    - Assume each line of code (LOC) = 10 bytes of code
    - Then each 100KB exe module = 10 KLOC
  - Each module has about 50 (to 500) bugs
  - So each host (node) has 150K (to 1.5M) bugs
- Same 150K bugs exist across much of the network
  - 150K bugs * 30K hosts = 4.5B bug instantiations in the network
- Further assumptions
  - Assume 10% of bugs can result in security failure
  - Assume 10% of those can be exercised remotely
  - Then there are 45M remotely exploitable security bug targets

Patching and Patch Management

The ultimate goal is to eliminate patching

Patching Challenges

Roadmap for the Evil-Doers

- Many of the products used by organizations that compose our critical infrastructure have required multiple patch releases for “critical” or “important” flaws in the past year.
- The size and complexity an institution increases the time required to remediate the vulnerability.
- Not patching quickly leaves the critical infrastructure at risk to exploit from vulnerabilities.
- The manpower to test and deploy these patches can cost an individual company millions of dollars annually.

Rapid Translation – Patch to Attack

Bad Guys Take a Lesson from the Antigen Community

- Snapshot of system exe pre-, post- patch
  - Find cause of vulnerability
- Automation to generate exploit code
  - Bad code available within days
- So patches have to be applied FAST
  - Which means automated patch management systems
- A fault in the automation can be disastrous
  - Infrastructure-wide collapse possible
- Bad-guy tools getting better
  - Automated virus from vulnerability < 1 year out?

Alternative Risk Mitigation Strategies

- Don’t expect to patch fast enough
  - Bad code at door well ahead of patch deployment
- Anticipate patch release cycles (predictability)
  - Bunch of code at pre-set intervals, including patches
- Oh oh! I can’t take down that system right now!
  - Patch on hand, bad code at door, what now?

Some Principles for Patch Management

- Patches must be well-tested, small, localized, reversible, installable without reboot, and provide common or consistent user experience.
- Also, features must not piggy-back on patches and “catching up” a system must be easy.
- Reboot issue is key: 10%-30% reduction (proposed by MSFT) is not enough, especially in service industry (Solaris is hot patchable)

Fixing and Solving

What Can We Do?

Process for Manufacturing Secure Code

- Methods for consistently secure software require
  - high level of security coding expertise
  - processes for producing secure software
  - adoption of processes, and
  - continuous use of learned secure coding expertise
- Low defect rate metrics may apply as measure
  - require strong motivators (and incentives)
- Must understand threats to systems in order to build secure software

Models For Understanding Threat

- Which approach is best?
  - Security code review
  - Penetration testing
  - Threat modeling
- Best to uncover issues in the design of the product early rather than trying to fix bugs late

Why Threat Modeling?

Alternatively, Extreme Programming

- You cannot build secure apps unless you understand threats
  - Adding security features does not mean you have secure software!
  - Can't call yourself secure just because you use SSL!
- Find issues before code is created
- Find different bugs than code review and testing
  - Implementation bugs vs. higher-level design issues

White Hat vs. Black Hat

(Different View of the Same Problem)

- White hat
  - Functionally, is the traffic adequately protected?
- Black hat
  - Can you force the traffic to be unprotected?
  - Is the crypto weak?
  - Where is/are the key(s) stored?
  - How are the keys exchanged?
  - Are the defense in depth methods ok?
  - Are there other conditions?

Threat Model Summary

- Scenario driven approach
- Determine privilege required to initiate data flow
  - Be wary of unauthenticated data flows
- All information disclosure threats are potentially privacy issues
- All non-mitigated threats are a potential vulnerability
- All security features must mitigate one or more threats

Motivations for Good, Bad, and Ugly

- Motivate development of more secure software during every phase of software development
  - Collective security is a societal problem
  - Secure code linked to developer performance
- Promote effective interaction between security researchers and software vendors
  - Managing disclosure for the win-win
- De-motivate malicious behavior
  - Multi-company program offering rewards for information leading to the conviction of cyber bad-guy

Code of Secure Code

- Are there are a limited number of well-defined, egregious security flaws that simply shouldn't be tolerated (and ought to be against the law)?
  - Product recalls for defects (and stupid mistakes)
  - Negative publicity for developers (vendors, in-house)
- Secure coding practices
  - Like fire code, building code
  - Accountability for failure to adhere to code (penalties)
- Product liability law for secure coding
  - Law or civil courts?
  - Flaws in software placed on the critical infrastructure

Recap and Summary

Why Security Problems With Software?

- Common development practices leave many vulnerabilities in developed software
  - Software supporting critical infrastructure (CI) cannot withstand attack
- Secure infrastructure → secure code
  - Code with vulnerabilities cannot live on the CI
- Must reduce specification, design, code defects
  - Defects are introduced by software developers
- Defect free code → retrained engineers

Improving Software Security & Safety

- a research and education issue for universities;
- a training issue for developers;
- a maintenance and patching issue for IT users;
- an ease-of use issue for users;
- a configuration issue for installers; and
- an enforcement issue for governments.

Increasing Software Security Means

- enhancing the education and training of developers to put security at the heart of software design and at the foundation of the development process;
- developing and sharing best practices to improve the quality of software, as well as the process so that systems are more resilient to attack;
- creating incentives that can create a culture of security awareness, and disincentives for malicious behavior; and
- making the patching process simple, easy, and reliable.

The Impact of Secure Software on Homeland Security

Ron Moritz, CISSP

Senior Vice President, Chief Security Strategist

Computer Associates *e*Trust™ Security Solutions

Ron.Moritz@CA.com