Information Warfare and Your Responsibilities as a Software Developer

Presented by
Mark Satterfield
Email: contact@marksatterfield.com
LinkedIn: http://www.linkedin.com/in/marksatterfield

Introduction to understanding hackers and protecting your software against attack

Outline

- Audience
- The law
  - Criminal law
  - EULA and Contract law
- Who and what are targets?
  - Type safe languages?
  - Type of applications?
- To stop an attack, think like an attacker
  - Why do attackers attack?
  - What hackers will do to your code (and what you should be doing!)
- A few of the common attack vectors and how to fix them
- Discussions and future reference
- Conclusions

Audience

This presentation is an **introduction** to secure software development concepts.

The presentation is particularly focused on ways that a typical hacker may look to compromise the Confidentiality, Integrity, and Availability triad.

We will **not** cover Computer Network Attack and Exploitation

Nothing novel! Everything in this presentation is assembled from open literature.

The Law

State criminal laws on hacking

- Florida Computer Crimes Act; Florida Statute 815
  - Attacks on computers or accessing computers without permission are in most cases felonies
  - Whoever willfully, knowingly, and without authorization destroys, takes, **injures**, or **damages** equipment or supplies used or intended to be used in a computer, computer system, or computer network; or whoever willfully, knowingly, and without authorization destroys, injures, or damages any computer, computer system, or computer network commits an offense against computer equipment or supplies
  - Whoever willfully, knowingly, and without authorization **accesses** or causes to be accessed any computer, computer system, or computer network; or whoever willfully, knowingly, and without authorization denies or **causes the denial** of computer system services to an authorized user of such computer system services, which, in whole or part, is owned by, under contract to, or operated for, on behalf of, or in conjunction with another commits an offense against computer users.

Federal criminal laws on hacking

- Digital Millennium Copyright Act
- Computer Fraud and Abuse Act
- USA PATRIOT Act
- Cyber Security Enhancement Act
- Economic Espionage Act
- Fraudulent Online Identity Sanctions Act
- Computer Software Privacy and Control Act
- Identity Theft Enforcement and Restitution Act
- Fraud and related activity in connection with computers (18 USC § 1030)

Civil laws on hacking

Most EULAs even prohibit reverse engineering

- Microsoft: "You may not reverse engineer, decompile, or disassemble the Software, except and only to the extent that such activity is expressly permitted by this EULA or applicable law notwithstanding this limitation"

The law?
Take away

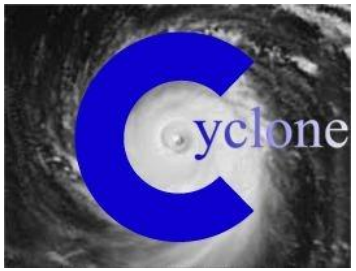
Just don’t do it. If you think you want to be a hacker, look for a job in:

- Ethical hacking
- Penetration testing
- Computer Network Operations

Who and what are targets

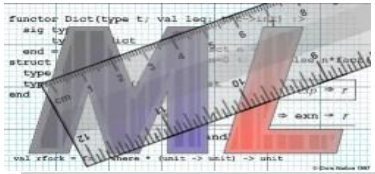
Do vulnerabilities exist in type safe languages?

- False sense of security - languages may be safe from buffer overflow, but not architectural vulnerabilities!
  - Integer overflow
  - Parameter tampering
  - Cross site scripting
  - Logic errors
  - Man in the Middle



"Java Still Not Safe, Security Experts Say"

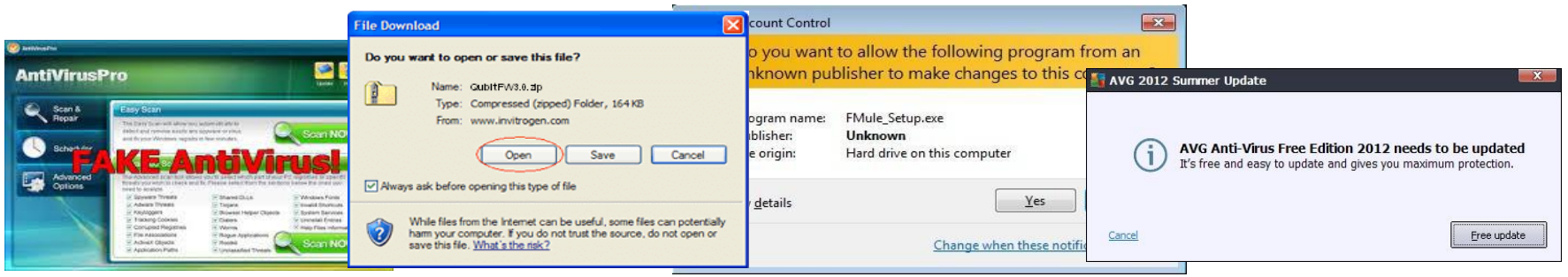
"Homeland Security warns Java still poses risks after security fix"



Do vulnerabilities exist in games and "reader" applications?

Yes, and serious ones! Just a few examples:

- Exploit game, then get users to request malware downloads via popup
  - "Free upgrade to full version now!"
  - These are not IE popups, these are program popups
- Arbitrary remote code execution
- Real life example: Java, Steam (game engine), Adobe PDF Reader, Microsoft Truetype font rendering engine



Who and what are targets?
Take away

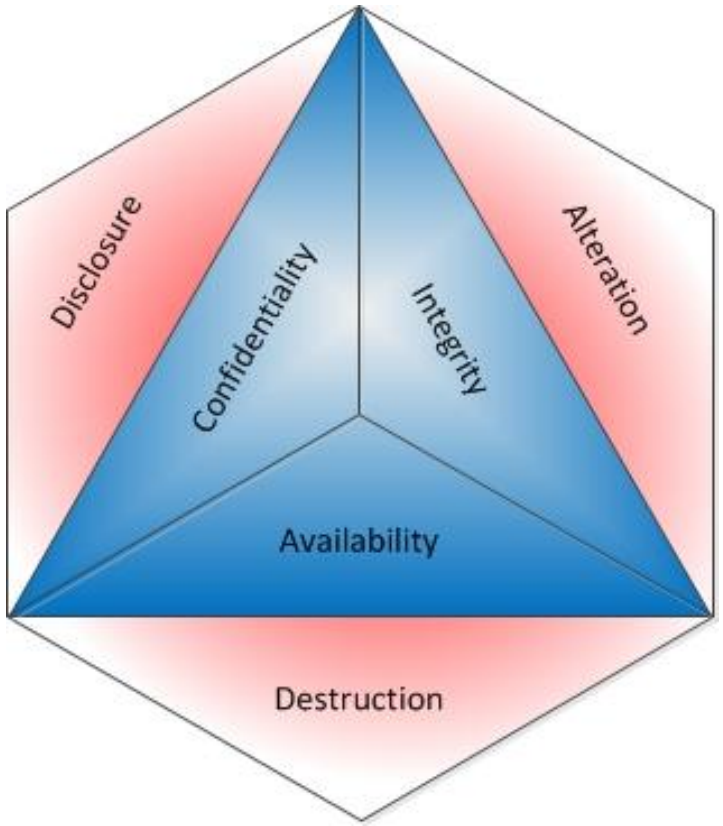
- All solutions
- All software
- All systems

To stop an attack, think like an attacker

“Think like a contrarian”

What is an attack?

- The Information Assurance triad (CIA)
  - Confidentiality
  - Integrity
  - Availability
- An attack is anything that compromises this triad (DAD)
  - Disclosure
  - Alteration
  - Destruction



Why do attackers attack?

- Money - Credit card info, DRM stripped material (DeCSS)
- Revenge - HBGary Federal, layoffs
- Fun - Morris Worm wanted to find out size of the Internet
- Politics - Information Operations including MILDEC, PSYOPS
- Chaos - DOS attacks... but DOS attacks might be a disguise. An attacker may have installed a rootkit, and needs to have the system rebooted

Definitions: Bug, Vulnerability, Exploit

- Bug, anomaly, defect, security flaw
  - Make the software work in an unintended way
- Vulnerability
  - "A flaw or weakness in a system's design, implementation, or operation and management that *could* be exploited to violate the system's security policy" (IETF RFC 2828)
- Exploit
  - A payload that exercises the vulnerability in a way that is useful for the attacker

How do hackers attack? First find a bug

- Find a bug, flaw, crash, or other anomalous behavior
  - Source code (white box) review if available
  - Decompiling, disassembling
    - Hex-Rays IDA
  - Identifying and reversing protocols
    - Wireshark
  - Automated fuzz testing (Mutation & Generation)
    - SDL MiniFuzz File Fuzzer (Microsoft)
    - Peach Fuzzer (peachfuzzer.com)
    - Radamsa (code.google.com/p/ouspg/wiki/Radamsa)

How do hackers attack?
Not finished yet ...

- Identify if the bug is a vulnerability
  - Most bugs do not expose a vulnerability
- Craft an exploit
  - and throw it at your box!

Think like a contrarian?
Take away

How to stop the attackers … ***Stop the bugs!***

But … is bug free code reasonable?

Maybe? We'll talk about this in a bit

Common attack vectors and how to fix them

Common attack vectors

- Data parsing attacks
  - Integer overflow
  - Buffer overflow
  - Protocol parsing
  - Uncontrolled format string
  - SQL Injection
- Data handling and logic attacks
  - Temporary files
  - Cookie poisoning
  - Forceful browsing
  - Volunteering too much information
  - Race conditions
- Transmission attacks
  - Man in the middle
  - Spoofing
  - DHCP/DNS attacks
- System attacks
  - Architectural attacks
  - Permissions & privesc
  - Default/weak/backdoor passwords
  - Scripts & macros
  - DLL replacement

Integer overflow (underflow)

- For a two byte integer
  - 32767 + 1 => -32768
- So why is this a problem?
  - Ex: shopping cart app. If user purchases one more than 32767 items, then winds up with a credit balance!
  - Ex: malloc(unsigned int). If assignment error passing a signed int to malloc, heap vulnerabilities can occur
- How to prevent these issues
  - Security/sanity checks. Be reasonable with user data bounds checks. Is it reasonable for a consumer to purchase a large number of an item?
  - Use appropriate variables, that is, use unsigned int where it is appropriate instead of signed int

Buffer overflow

- In some circles, considered a "classic" attack vector
  - Code Red, MS-Blaster
- So why is this a problem?
  - Works by overflowing a stack or heap buffer
  - Overwrite a return address and the attacker may be able to execute arbitrary code
- How to prevent these issues
  - White box - Replace vulnerable routines with overflow safe routines >> strcpy(), gets(), strcat(), sprintf()
  - Validate all inputs -- don't trust data!
    - Network, API, Filesystem, User Inputs

Data format and protocol parsing

- Encoding and decoding data is complicated
- Commercial known vulnerabilities
  - Microsoft's TCP/IP implementation (denial of service and remote code execution vulnerabilities)
  - MySQL authentication bypass
- How to avoid this issue
  - Avoid complicated proprietary formats - even though commercial products have vulnerabilities, reinventing a parser is not likely to solve the problem
  - Use parsing libraries
  - XML
  - Database engines (MySQL)

SQL Injection

Allows arbitrary SQL command execution

- cmd = "SELECT * FROM users WHERE name = '" + str + "';"
  Should result in
  >> SELECT * FROM users WHERE name = 'myname';
- but if str = "a'; SELECT * FROM userinfo WHERE 't' = 't" then command becomes
  >> SELECT * FROM users WHERE name = 'a'; SELECT * FROM userinfo WHERE 't' = 't';
  - The first select does nothing. The second command selects everything from the userinfo table!
- How to prevent?
  - Parameterized statements, Escape characters, Pattern checks

Temporary file compromise

- Example: A player that plays DRM protected video
  - Created for a system with not enough horsepower for simultaneous decrypting and playing
  - Solution? Write plaintext to a temp file, then play from temp file
- Compromise is: Malicious user can grab the temp file, then has an unprotected version of the protected file!
- Prevention
  - Avoid temporary files holding plaintext protected data
  - Be aware of paging and swap file use
  - If absolutely necessary, "read" protect the file

Volunteering too much information

- Responding with "Password incorrect for this username" or "Authentication credentials are invalid"
  - An attacker now knows the username is valid!
  - Dictionary or other attack ensues
- Another common response - when user asks for password info, the system responds "invalid username"
  - Now malevolent user can determine if a username really does exist
- How to avoid?
  - Avoid providing too much information
  - Send an email with registration info if not registered
  - It is a careful balancing act between a smooth user experience and publishing a vulnerability

Man in the Middle attacks

- Your commerce site clients may use public wifi
  - To the site, attacker masquerades as the customer
  - To the customer, attacker masquerades as the site
- Is this your responsibility?
  - You bet! You are building a commerce site, it is your responsibility to protect your clients
- This is an architectural issue
  - Code analysis is not going to uncover a MITM vuln
- To mitigate
  - SSL certificates
  - Public Key Infrastructure

Default/weak/backdoor passwords

- Systems are often deployed with default passwords. Some vendors install sentinel passwords for manufacturing and deployment use
  - Seriously? Yes, seriously.
  - Regarding Barracuda Networks, "*Several undocumented operating system user accounts exist on the appliance. They can be used to gain access to the appliance via the terminal but also via SSH.These accounts are undocumented and can not be disabled.*"
- How to check
  - Disassembler or debugging tool such as Hexrays IDA
- How to avoid: Just say no
  - Force user to create a password during initialization
  - Force strong passwords to prevent dictionary attacks

Scripts & Macros

- Examples: Melissa (VB Macro); "I Love You" virus; Browser phishing attacks; Cross site scripting attacks
- Code execution
  - Could cause a Flash file to execute, which may have a vulnerability
  - Could destroy files on local machine
  - Pop ups
- Local access
  - Now the user is on the local machine, can he hack the router or other systems? Now a LAN attack!
  - Can the attacker write a DLL to the local directory that will provide a system level attack?

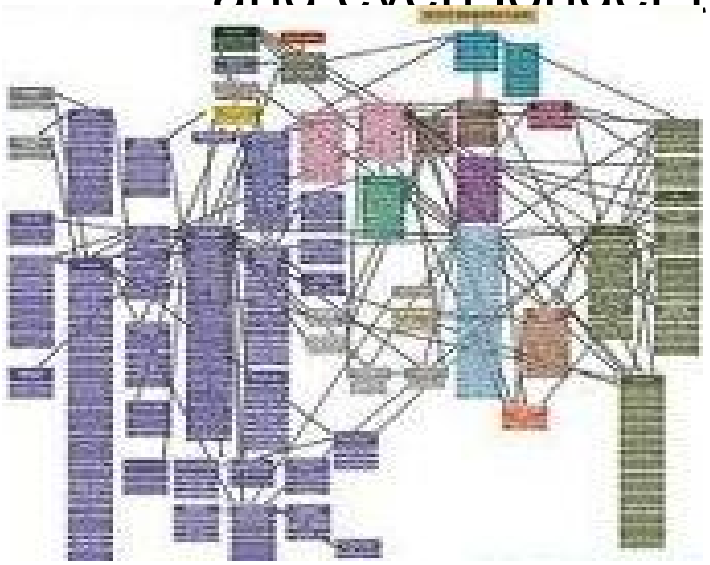
Common attack vectors? Take away

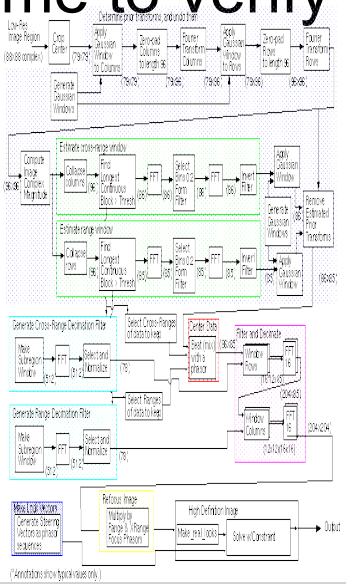
- Way too many to enumerate
  - Secure Operating Systems? Still have software architecture issues
  - Type safe languages? Still have architecture issues
- Our job as software developers is, be aware

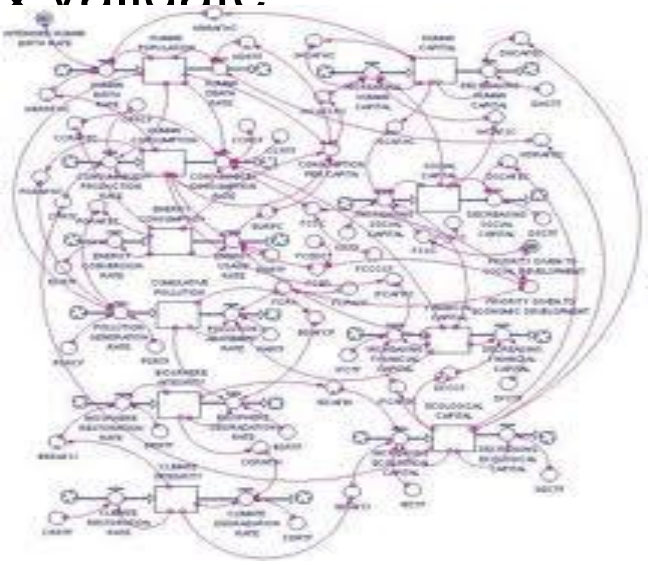
Discussions and future reference

With all this knowledge
Why do vulnerabilities *still* exist?

- Software is complex, systems are even more complex
  - Secure software design is hard
  - Secure validation is even harder if not impossible
- Time to market restrictions
  - "Perfect" software takes a long time to implement, and even longer time to verify & validate







Is there resistance to "safer" or more secure coding?

- For software developer, no incentive for safer code
  - Timeline & schedule pressures dominate
  - *"If my manager is measuring my evaluations and raises based on SLOC, by golly I'll give him SLOC!"*
  - Changes need to start at the top before those changes can have any impact for the individual contributor
- For company, no incentive to invest in secure code
  - Depending on software, how can it be marketed?
  - More testing = longer schedules and possibly lost market opportunity
  - More cost, delayed schedule... is it worth it?

So then why perform any work towards "safer" coding practices?

- Your users and customers will appreciate the extra work
  - This includes coding, design, architecture, testing... everything works together to create good code
  - Without a vulnerability, there is no exploit!
  - Bad "exploitable code" publicity may compromise your company
    - One of the reasons Chrome became popular
- Sometimes compliance and fear of lawsuits
  - HIPAA, SEC

Okay we'll do it... but how?

- How to save your code from vulnerabilities?
  - Be familiar
  - Training
  - Test ... test ... test !
    - Perform tests early and often
    - Verification & Validation are not enough
    - Conformance testing, certifications
  - And again, be familiar
    - As you get more familiar with software security, you will become more aware of what to test

What can you do as a developer Static analysis

- Code walkthroughs
  - Make your coding practices easy to understand
  - If during a code review a question arises, then maybe the code is too complex
- Static analysis tools
  - Usually programming language specific
  - cppcheck, Protecode, splint, FindBugs (Java)

What can you do as a developer
Static analysis in reverse

- Always a good idea to use a disassembler on your own code
  - For example, Hex-Rays IDA
- Why? Insecure Compiler Optimization
  Sometimes the compiler messes things up. With a disassembler, you can see in a high level language what your exe looks like
  - Dead store failure
  - Wraparound checks

What can you do as a developer Dynamic analysis

- Unit testing & Systems testing
  - Regression & non-regression testing
  - Perform testing early and often
- Debuggers
  - OllyDbg, gdb, useful for testing

Tons of tools - use them!

- Network & systems testing
  - Nessus (tenable)
  - Retina (eEye, now beyondtrust)
  - MS Baseline Security Analyzer (Microsoft)
  - Metasploit (RAPID7)
- Static analysis scanners
  - Security AppScan Source (IBM)
  - Flawfinder (Dwheeler)
  - FindBugs (findbugs.sourceforge.net)
  - RATS (Secure Software Inc)
  - Owasp Orizon (owasp.org)
- Disassembly & debugging
  - IDA - debugger & disassembler (hex-rays)
  - OllyDbg (ollydbg.de)
  - gdb (GNU Debugger)
- Password "auditors"
  - LC4 (@stakes)
  - John the Ripper (Openwall)
  - L0phtcrack (L0pht Heavy Industries)

Safe coding practices & Development Rules of thumb

- Complexity is the enemy of security
  - Keep your code as simple as possible
  - Avoid obscure code and undefined behavior
- Use minimal privileges for deployed applications
  - Don't require user to have root priv if not strictly required
- Catch all bugs and questionable results
  - Your software needs to catch anomalies
  - Test & apply tools early and often
  - Protect at the unit level, plus protect again anywhere you like (like in the client side browser), but keep the unit level protections (e.g., don't trust the user!)

Safe coding practices & Development rules of thumb

- Never trust the client nor user
  - nor network nor file system nor DLLs nor cookies nor anything else that is not part of your executable, and sometimes not even that (hacker could nop the authentication routine)
  - This includes both inputs and outputs
    - Perform sanity checks on server side, not client
    - Don't volunteer too much information
- Expect adversity, even if your program is simple
  - Your program may simply be the vector into the intended system

Conclusion

Conclusion
When is security testing complete?

Kind of ... well... never.

Once an application is deployed, REAL testing begins

- This presentation was to create awareness and motivate the developer to further study
- As you become more aware of vulnerabilities, retest
  - Never assume a code set is bug free
  - Sometimes you won't know about a vulnerability until your customers have been compromised. Be ready by creating a plan on how to handle it.
- When deploying a "new version", realize that hackers are going to look at the 0days... and attack the base of users who haven't updated

And just remember…

"Hey, let's be careful out there."

END

References
Certifications & testing

- Personnel certifications
  - GIAC Secure Software Programmer (GSSP), http://software-security.sans.org
  - Certified Information Systems Security Professional (CISSP), http://www.isc2.org
- Conformance testing, Source Code Analysis Laboratory (SCALe), http://www.cert.org/secure-coding/scale/
- Bugtraq (www.securityfocus.com)
- Common Vulnerability & Exposure Database (cve.mitre.org)
- www.cert.org
- www.defcon.org
- www.blackhat.com
- www.rsaconference.com
- Webinars (www.sei.cmu.edu/library/webinars.cfm)
- Introduction to Secure Coding Guide (developer.apple.com)