

Software Assurance Tool Status and Gaps

Dr. Larry Wagoner
National Security Agency
301-688-2827
l.wagone@radium.ncsc.mil

Outline

- Threat
- Scope/Focus
- Prevention
- Detection
- Reaction

The Threat

- There is ample opportunity to implant malicious code
- Malicious coder could be anyone, anywhere
- Adversaries can have a lot of patience and can change over time
- Malicious implant may be only one part of an attack
- Software can be attacked or exploited at any point in its life-cycle: development, distribution, operational use, maintenance
- Software can be attacked cheaply and with a low level of risk

Software Assurance (SwA)

Goal: Eliminate malicious code and reduce software vulnerabilities

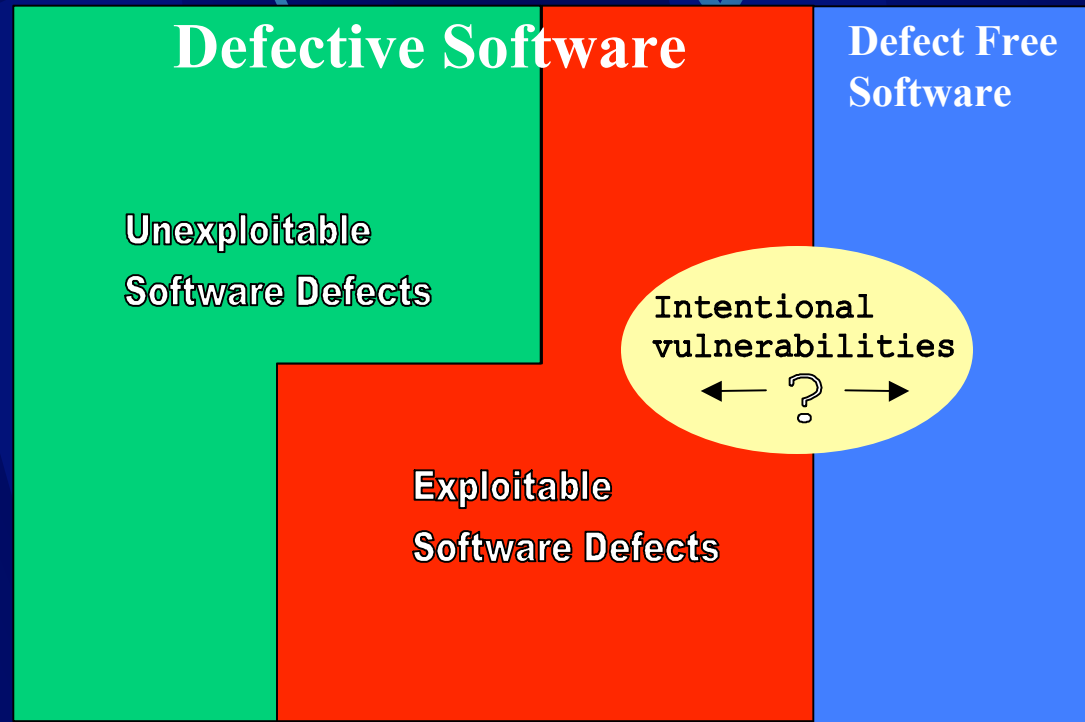
- λ Trojans/back doors
- λ Time and logic bombs
- λ Exfiltration

Software Assurance: “The level of confidence that software is free of vulnerabilities, either intentionally or unintentionally designed or inserted during software development and/or the entire software lifecycle.”



SwA scope is the red and yellow areas

**S
o
f
t
w
a
r
e**



- Not all defects are exploitable vulnerabilities and not all vulnerabilities are defects

Focus

- Focus started with intentional vulnerabilities
 - λ Hardest of a hard problem
 - λ Vulnerability could be disguised
 - λ as a feature
 - λ as an unintentional vulnerability
 - λ anywhere in the code
 - λ not just security features
 - λ time bomb/logic bomb
- Could be inserted at any point in the lifecycle
- Company could be the instigator or could be a victim also

Principal Goals

- Maintain system availability and predictability
 - λ No DOS
 - λ Timely and reliable access to systems
- Protect Intellectual Property
 - λ No exfiltration
 - λ Information not disclosed to anyone unauthorized
- Ensure data integrity
 - λ Protection against unauthorized modification or destruction of data
 - λ Lose this and the data is worthless

Past Solution

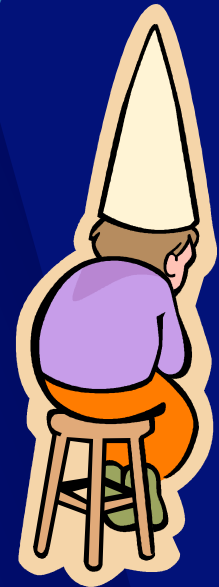
- Air-gapped systems
 - λ Doesn't fully protect against data alteration or system availability
 - λ Hard to work in a vacuum – but connectivity to other organizations leads to potential problems
 - λ Everyone wants to and needs to be interconnected
 - λ Except in very rare instances, not a solution anymore

Prevention

- Best way to fix problems
- Have known how to prevent for a long time, yet developers still make mistakes
- Little control or access into globally produced software

Prevention Needs

- Awareness
- Truly trustworthy computing base
- Need a business case to sell prevention
- New standards to protect developers from themselves
- Compose secure systems from independent secure components
- Develop more cost-effective methods for high assurance software development (and in general for low and medium assurance)
- Input Validation Standards
- Improved compilers



Improved Security through Compilers - Specifics

- Microsoft – deprecation of the some of the roots of buffer overflows in C and C++ in Visual Studio 2005
 - λ string.h
 - λ Need to phase out or make `_CRT_SECURE_NO_DEPRECATED` more painful to use
 - λ Need this trend in other compilers

Detection

- Looking for a needle in a haystack
- Gray area between features and vulnerabilities
- Trust, but verify
- Important defense for malicious vulnerabilities
- Don't always want to "tip your hand" that you're examining a particular product



Detection Needs

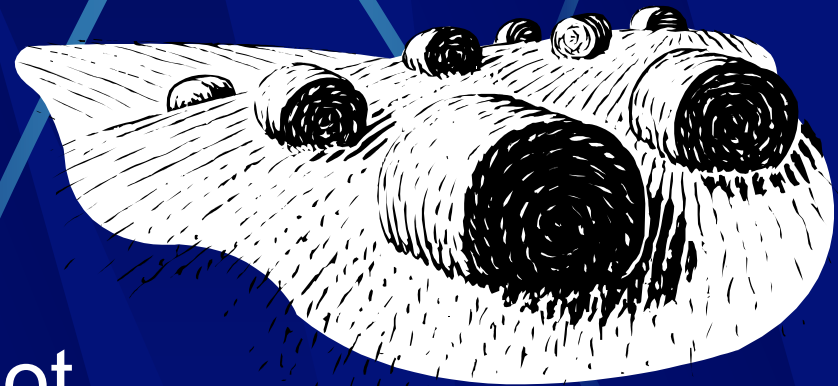
- Need to move past evaluations measured in man-months as in “I had someone look at it for three months straight, so it must be good”
- Move toward very predictable/recreatable analysis
- Quick and scalable analysis - we can buy supercomputers if needed
- Metrics to measure “assuredness”
- Evaluations for software being used in specific applications

Detection Needs

- Improved binary and source scanning tools
- Dynamic vs. static analysis
- All tool categories have value
- Perfect tools?
 - λ Value of tools can be rapidly diminished by too many false positives
 - λ Would prefer easy to use tools with near zero false positives so developers will use the tools
 - λ Don't need to be "perfect"
 - λ Incremental approach
- Need to independently verify claims

Reaction

- Looking for a needle in a haystack
- Currently overwhelming admins
- Discovery is too late – damage is done
- Forensics takes a lot of resources – almost a luxury in a resource constrained environment



Reaction Needs

- Methods to minimize/control the functionality of products
- Mechanisms to detect or counter runtime exploits
- Reaction, by human nature, has good market demand
- Damage control

Summary

- Considerable opportunities to insert vulnerabilities
- Software assurance is a hard and challenging problem
- Perfection is not needed initially
- Incremental improvements
- Need predictable and scalable analysis tools that increase trust in software

The End.

Dr. Larry Wagoner

National Security Agency

301-688-2827

l.wagone@radium.ncsc.mil



Backup

The Farewell Dossier

- Soviets were stealing large amounts of Western technology in the late 1970's/early 1980's
- CIA and DoD modified products were “made available”
 - λ Contrived computer chips found their way into Soviet military equipment
 - λ Defective plans disrupted the output of chemical plants and a tractor factory
 - λ Flawed turbines were installed on a gas pipeline
- Soviets were left to wonder what else was “customized”