Ground System Attacks

Jared Ablon (jablon@mitre.org)

February 25, 2014

Introduction to Ground System Software Assurance

MITRE

Historical Evidence:

Chinese Military Suspected In U.S. Ground Station Hacking (2007)

"Computer hackers, possibly from the Chinese military, interfered with two U.S. government satellites four times in 2007 and 2008 through a ground station in Norway…"

Landsat-7 and Terra AM-1 satellites use the Svalbard Satellite Station in Norway that "routinely relies on the Internet for data access and file transfers,"

"Chinese Military Suspected in Hacker Attacks on U.S. Satellites", Bloomberg, http://www.bloomberg.com/news/2011-10-27/chinese-military-suspected-in-hacker-attacks-on-u-s-satellites.html

MITRE

Historical Evidence:

Drone Ground System's Closed Network Hacked (Oct 2011)

- Key Logger infects Predator and Reaper drones' ground station cockpits
- "…they're sure that the infection has hit both classified and unclassified machines at Creech… secret data may have been captured by the key logger, and then transmitted over the public internet to someone outside the military chain of command."

Wired Magazine: http://www.wired.com/dangerroom/2011/10/virus-hits-drone-fleet/

MITRE

Historical Evidence:

Zombie Apocalypse Starts in Montana (Feb 2013)

- Emergency Broadcast System Hacked and broadcast a zombie apocalypse alert
- "Civil authorities in your area have reported that the bodies of the dead are rising from their graves and attacking the living... Do not attempt to approach or apprehend these bodies as they are considered extremely dangerous."

- "…it inspired several calls to local police from concerned citizens who believed it to be real"

http://www.informationweek.com/security/attacks/zombie-alert-hoax-emergency-broadcast-sy/240148355
http://www.huffingtonpost.com/2013/02/11/krtv-fake-zombie-alert_n_2665469.html

MITRE

Why Hack A Ground System:

It May be the Easiest Target

- **Exploiting satellites can be tough**
  - Physical access is a challenge
  - Fuzzing RF is noisy and difficult for a live system
- **Exploiting receiver equipment most likely not as effective**
  - Not scalable
  - Difficult/Impossible to affect satellites or other receiver equipment

- **Why is a ground system an attractive target?**
  - Scalable in terms of affecting multiple receivers
  - Physical access may be possible
  - Network access may be possible
  - Can affect/exploit satellites without physical access

My Ground System Isn't On the Internet:

It Is Still Vulnerable?

- **Other attack vectors**
  - Social Engineering
  - Insider Threat
  - Closed Network Access
  - Thumb Drive Attack (STUXNET)
  - Sound waves (BadBIOS)
  - Physical Access
  - RF Based Attack
  - Etc.

- **Most of these rely on software vulnerabilities to degrade, disrupt, deny, or destroy**

MITRE

How Hacking Works:

Degrade, Disrupt, Deny, Destroy

- **Most current Cyber incidents are initiated with software vulnerabilities (social engineering/insider attacks excluded)**
- **Ground Systems are becoming more software intensive systems**
- **Software vulnerabilities could enable**
  - Complete (root level) control of the system
  - Unauthorized (user level) access to the system
  - Persistent access
  - Installation of
    - Malware
    - Key loggers
    - Back doors
    - Data exfiltration tools
    - Computer Network Attack tools

What is Software Assurance?

Software Assurance (SwA) is the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its life cycle, and that the software functions in the intended manner.

– CNSS 4009, National IA Glossary

Software Assurance is a set of **systematic activities** to increase the **trustworthiness** of software so as to assure its **predictable execution**.

Why implement Software Assurance:

Assured Software …

- Does what it should
- Doesn't do what it shouldn't
- Is of high quality
- Is reliable
- Resists natural adversity
- Resilient to intelligent adversaries

Software Assurance:

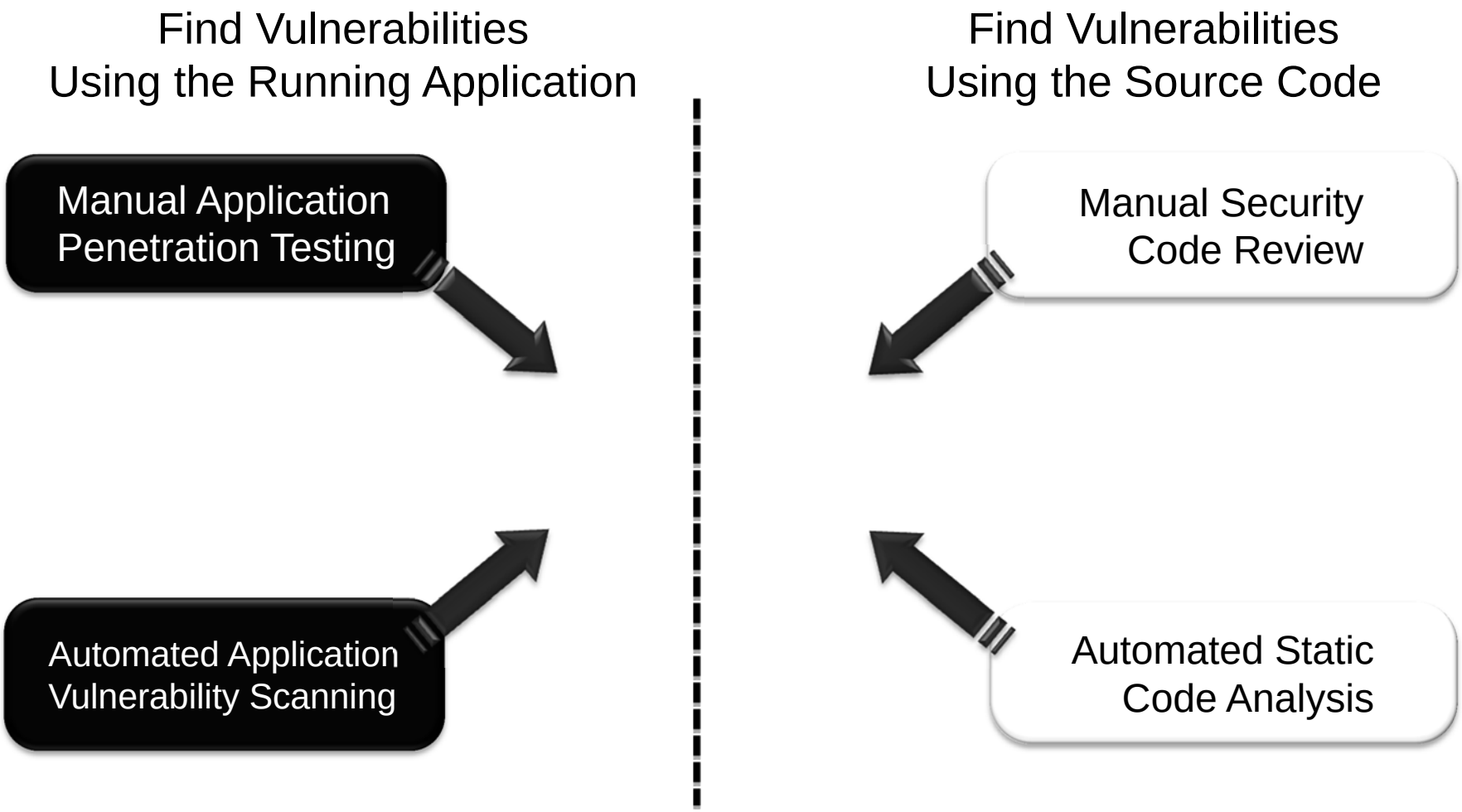
How to Implement on a Ground System

- **Require that developers follow Secure Coding Standards**
  - DISA's Application Development & Security, Security Technical Implementation Guide (STIG) is a good start
    - Validate all input
    - Use secure functions (memcpy_s, etc.)
    - Use Prepared Statements (Parameterized Queries)
    - Etc.
  - Train developers on the standards
- **Educate developers on the OWASP Top 10 and SANS Top 25**
  - Include mitigation techniques
- **Apply OWASP's Application Security Verification Techniques**
  - Static/Dynamic testing should be part of the software development life cycle
  - Remediation procedures and strategy should be put in place

OWASP Top 10, https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
CWE/SANS Top 25, http://www.sans.org/top25-software-errors

MITRE

OWASP Application Security Verification Techniques

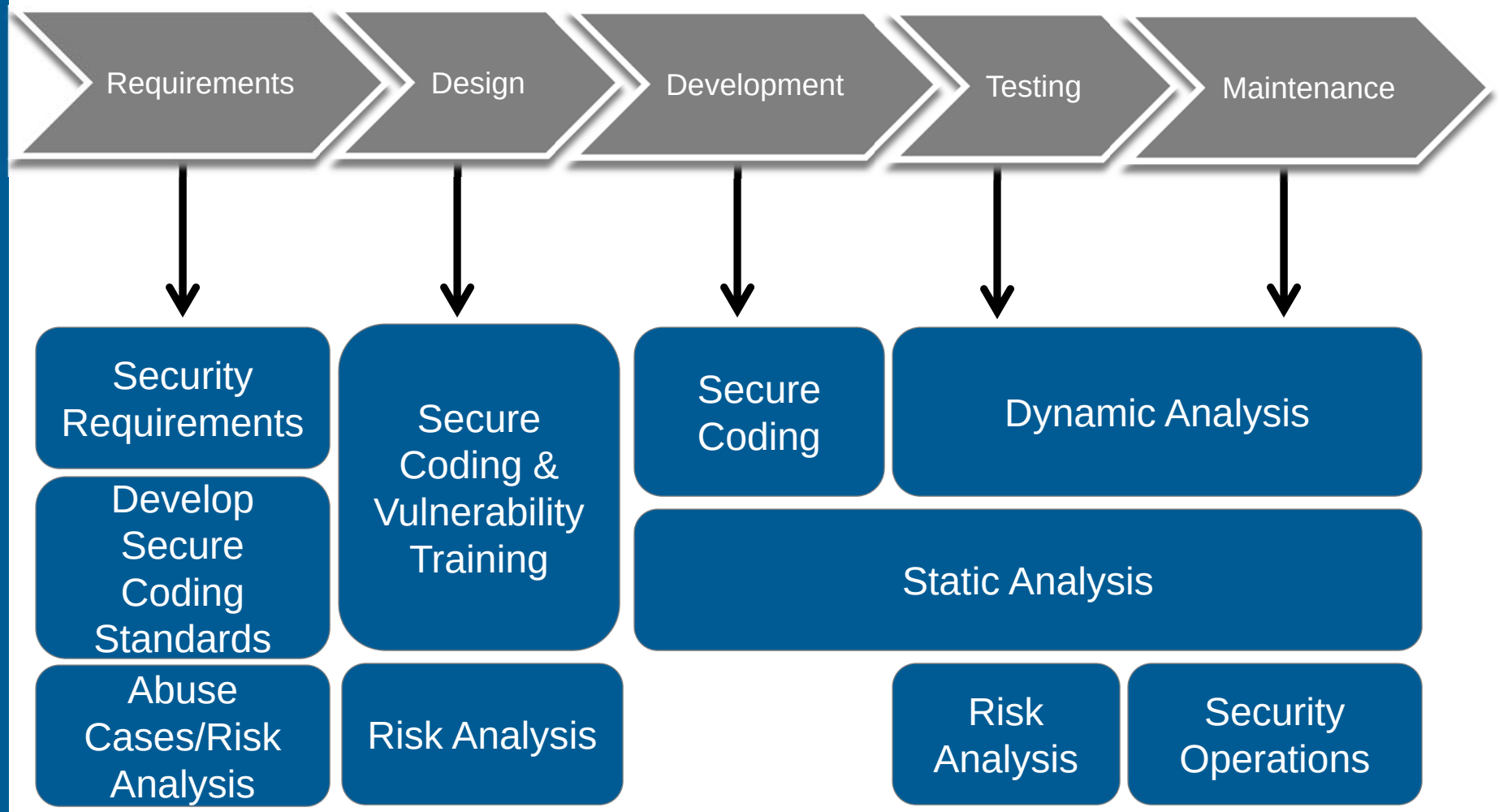


OWASP Application Security Verification Standard (ASVS), Mike Boberski, Jeff Williams, Dave Wichers, https://www.owasp.org/images/a/a0/Wichers_-_About_OWASP_ASVS_Web_Edition_v2.pdf

MITRE

Software Development Life Cycle (Model Agnostic):

A Software Assurance Approach



Reference: “Software Security Touchpoint: Architectural Risk Analysis”, Gary McGraw, http://www.cigital.com/presentations/ARA10.pdf

MITRE

Additional Software Hacking Vectors:

Ground System use of Legacy Software and FOSS

- **Heavy Reliance on Legacy Software & Free and Open Source Software (FOSS)**
  - FOSS may not save time/money if integrated securely
    - Maintenance tail is large
    - Secure Code compliance is difficult
    - Implementing Application Security Verification Techniques and a remediation strategy is cumbersome
      - Developers do not “own” the code so remediation takes longer
- **Commercial Off The Shelf (COTS) Software**
  - Security is more difficult to control
  - Source code is not available

FOSS as an Attack Vector:

Why Is Using FOSS Risky?

- **Security is not a priority with most FOSS packages**
  - Even though code is available, how many people really review it?
- **Source code is available for anyone to find vulnerabilities**
  - Current vulnerabilities are often listed on the FOSS website
- **Community support cannot be relied upon for patches/updates**
  - If you fix a vulnerability, what happens with a new release?
- **Community support today does not mean there will be support tomorrow**

MITRE

Defense in Depth:

Additional Avenues for Reducing Risk

- **Do not install software that is not necessary**
  - Minimize the attack surface
- **Data Execution Prevention (DEP)**
  - Intended to prevent executing code from a non-executable region in memory
- **Address Space Layout Randomization (ASLR)**
  - Randomly arranges data areas of a program in memory to help prevent reliable jumping to a particular area
- **Firewalls, Intrusion Detection & Protection Systems**
  - These may not necessarily prevent software attacks, but do make it harder to go unnoticed

Conclusions:

Software will never be 100% vulnerability free

- **Implement Software Assurance practices to minimize vulnerabilities and reduce risk**
- **Reduce reliance on legacy software, FOSS, and COTS**
  - Include them under Software Assurance process
- **Patch all FOSS/COTS and run Dynamic and Static tools against them to test for vulnerabilities**
  - Be prepared to deal with what you find
- **Add Defense in Depth (Data Execution Prevention, Address Space Layout Randomization, Firewalls, etc)**
- **Nothing is foolproof**
  - Decrease the attack surface
  - Increase the cost (time and money) for an attacker

Questions?

MITRE