Lecture 18
November 6, 2023

Formal Verification

- Mathematically verifying that a system satisfies certain constraints
- *Preconditions* state assumptions about the system
- *Postconditions* are result of applying system operations to preconditions, inputs
- Required: postconditions satisfy constraints

Penetration Testing

- Testing to verify that a system satisfies certain constraints

- Hypothesis stating system characteristics, environment, and state relevant to vulnerability

- Result is compromised system state

- Apply tests to try to move system from state in hypothesis to compromised system state

Notes

- Penetration testing is a *testing* technique, not a verification technique
  - It can prove the *presence* of vulnerabilities, but not the *absence* of vulnerabilities
- For formal verification to prove absence, proof and preconditions must include *all* external factors
  - Realistically, formal verification proves absence of flaws within a particular program, design, or environment and not the absence of flaws in a computer system (think incorrect configurations, etc.)

Penetration Studies

- Test for evaluating the strengths and effectiveness of all security controls on system
  - Also called *tiger team attack* or *red team attack*
  - Goal: violate site security policy
  - Not a replacement for careful design, implementation, and structured testing
  - Tests system *in toto*, once it is in place
    - Includes procedural, operational controls as well as technological ones

Goals

- Attempt to violate specific constraints in security and/or integrity policy
  - Implies metric for determining success
  - Must be well-defined
- Example: subsystem designed to allow owner to require others to give password before accessing file (i.e., password protect files)
  - Goal: test this control
  - Metric: did testers get access either without a password or by gaining unauthorized access to a password?

Goals

- Find some number of vulnerabilities, or vulnerabilities within a period of time
  - If vulnerabilities categorized and studied, can draw conclusions about care taken in design, implementation, and operation
  - Otherwise, list helpful in closing holes but not more
- Example: vendor gets confidential documents, 30 days later publishes them on web
  - Goal: obtain access to such a file; you have 30 days
  - Alternate goal: gain access to files; no time limit (a Trojan horse would give access for over 30 days)

Layering of Tests

1. External attacker with no knowledge of system

    -   Locate system, learn enough to be able to access it

2. External attacker with access to system

    -   Can log in, or access network servers
    -   Often try to expand level of access

3. Internal attacker with access to system

    -   Testers are authorized users with restricted accounts (like ordinary users)
    -   Typical goal is to gain unauthorized privileges or information

Layering of Tests (con't)

- Studies conducted from attacker's point of view

- Environment is that in which attacker would function

- If information about a particular layer irrelevant, layer can be skipped
  - Example: penetration testing during design, development skips layer 1
  - Example: penetration test on system with guest account usually skips layer 2

Methodology

- Usefulness of penetration study comes from documentation, conclusions
  - Indicates whether flaws are endemic or not
  - It does not come from success or failure of attempted penetration
- Degree of penetration's success also a factor
  - In some situations, obtaining access to unprivileged account may be less successful than obtaining access to privileged account

Flaw Hypothesis Methodology

1. Information gathering
   - Become familiar with system's functioning
2. Flaw hypothesis
   - Draw on knowledge to hypothesize vulnerabilities
3. Flaw testing
   - Test them out
4. Flaw generalization
   - Generalize vulnerability to find others like it
5. (*maybe*) Flaw elimination
   - Testers eliminate the flaw (usually *not* included)

Information Gathering

- Devise model of system and/or components
  - Look for discrepancies in components
  - Consider interfaces among components

- Need to know system well (or learn quickly!)
  - Design documents, manuals help
    - Unclear specifications often misinterpreted, or interpreted differently by different people
  - Look at how system manages privileged users

Flaw Hypothesizing

- Examine policies, procedures
  - May be inconsistencies to exploit
  - May be consistent, but inconsistent with design or implementation
  - May not be followed
- Examine implementations
  - Use models of vulnerabilities to help locate potential problems
  - Use manuals; try exceeding limits and restrictions; try omitting steps in procedures

Flaw Hypothesizing (*con't*)

- Identify structures, mechanisms controlling system
    - These are what attackers will use
    - Environment in which they work, and were built, may have introduced errors
- Throughout, draw on knowledge of other systems with similarities
    - Which means they may have similar vulnerabilities
- Result is list of possible flaws

Flaw Testing

- Figure out order to test potential flaws
  - Priority is function of goals
    - Example: to find major design or implementation problems, focus on potential system critical flaws
    - Example: to find vulnerability to outside attackers, focus on external access protocols and programs

- Figure out how to test potential flaws
  - Best way: demonstrate from the analysis
    - Common when flaw arises from faulty spec, design, or operation
  - Otherwise, must try to exploit it

Flaw Testing (*con't*)

- Design test to be least intrusive as possible
  - Must understand exactly why flaw might arise
- Procedure
  - Back up system
  - Verify system configured to allow exploit
    - Take notes of requirements for detecting flaw
  - Verify existence of flaw
    - May or may not require exploiting the flaw
    - Make test as simple as possible, but success must be convincing
  - Must be able to repeat test successfully

Flaw Generalization

- As tests succeed, classes of flaws emerge
  - Example: programs read input into buffer on stack, leading to buffer overflow attack; others copy command line arguments into buffer on stack $\Rightarrow$ these are vulnerable too

- Sometimes two different flaws may combine for devastating attack
  - Example: flaw 1 gives external attacker access to unprivileged account on system; second flaw allows any user on that system to gain full privileges $\Rightarrow$ any external attacker can get full privileges

Flaw Elimination

- Usually not included as testers are not best folks to fix this
  - Designers and implementers are
- Requires understanding of context, details of flaw including environment, and possibly exploit
  - Design flaw uncovered during development can be corrected and parts of implementation redone
    - Don't need to know how exploit works
  - Design flaw uncovered at production site may not be corrected fast enough to prevent exploitation
    - So need to know how exploit works

Versions

- These supply details the Flaw Hypothesis Methodology omits
- Information Systems Security Assessment Framework (ISSAF)
  - Developed by Open Information Systems Security Group
- Open Source Security Testing Methodology Manual (OSSTMM)
- Guide to Information Security Testing and Assessment (GISTA)
  - Developed by National Institute for Standards and Technology (NIST)
- Penetration Testing Execution Standard

ISSAF

- Three main steps
  - *Planning and Preparation Step*: sets up test, including legal, contractual bases for it; this includes establishing goals, limits of test
  - *Assessment Phase*: gather information, penetrate systems, find other flaws, compromise remote entities, maintain access, and cover tracks
  - *Reporting and Cleaning Up*: write report, purge system of all attack tools, detritus, any other artifacts used or created

- Strength: clear, intuitive structure guiding assessment

- Weakness: lack of emphasis on generalizing new vulnerabilities from existing ones

OSSTMM

- Scope is 3 classes
  - *COMSEC*: communications security class
  - *PHYSSEC*: physical security class
  - *SPECSEC*: spectrum security class
- Each class has 5 channels:
  - *Human channel*: human elements of communication
  - *Physical channel*: physical aspects of security for the class
  - *Wireless communications channel*: communications, signals, emanations occurring throughout electromagnetic spectrum
  - *Data networks channel*: all wired networks where interaction takes place over cables and wired network lines
  - *Telecommunication channel*: all telecommunication networks where interaction takes place over telephone or telephone-like networks

OSSTMM (con't)

- 17 modules to analyze each channel, divided into 4 phases
  - *Induction*: provides legal information, resulting technical restrictions
  - *Interaction*: test scope, relationships among its components
  - *Inquest*: testers uncover specific information about system
  - *Intervention*: tests specific targets, trying to compromise them
  These feed back into one another
- Strength: organization of resources, environmental considerations into classes, channels, modules, phases
- Weakness: lack of emphasis on generalizing new vulnerabilities from existing ones

GISTA

- GISTA has 4 phases:
  - *Planning*, in which testers, management agree on rules, goals
  - *Discovery*, in which testers search system to gather information (especially identifying and examining targets) and hypothesizing vulnerabilities
  - *Attack*, in which testers see whether hypotheses can be exploited; any information learned fed back to discovery phase for more hypothesizing
  - *Reporting*, done in parallel with other phases, in which testers create a report describing what was found and how to mitigate the problems
- Strength: feedback between discovery and attack phases
- Weakness: quite generic, does not provide same discipline of guidance as others

PTES

- 7 phases
  - *Pre-engagement interaction*: testers, clients agree on scope of test, terms, goals
  - *Intelligence gathering*: testers identify potential targets by examining system, public information
  - *Thread modeling*: testers analyze threats, hypothesize vulnerabilities
  - *Vulnerability analysis*: testers determine which of hypothesized vulnerabilities exist
  - *Exploitation*: testers determine whether identified vulnerabilities can be exploited (using social engineering as well as technical means)
  - *Post-exploitation*: analyze effects of successful exploitations; try to conceal exploitations
  - *Reporting*: document actions, results
- Strengths: detailed description of methodology
- Weakness: lack of emphasis on generalizing new vulnerabilities from existing ones