Model Checking To Analyze Network Vulnerabilities

Ronald W. Ritchey
ritchey_ronald@bah.com

Paul Ammann
pammann@gmu.edu

Introduction

- Combining services may result in vulnerability
  - Example: (ftp + http) hosted on same machine

- Many tools to check host configuration vulnerabilities
  - Example: COPS, Cyber Cop, System Scanner...
  - Good for checking host vulnerabilities but not look for combinations of configurations on the same host or between hosts.

Introduction (Cont.)

- To view overall security of network
  - Vulnerabilities on single host + relationships between hosts on network
  - NetKuang: search algorithm to identify vulnerabilities

- This paper go for modeling based approach.

Model Checking

- Model Checking specification has two parts
  - Model
  - Checker

Model Checking Specification

- Model
  - State Machine defined in terms of
    - Variables
    - Initial values for the variables
    - Conditions for variables to change values

- Temporal Logic Constraints over states and execution paths

Model Checker

- Visit all reachable states
- Verify logical constraints over each path
- Provide counterexample (sequence of events)
SMV Model Checking Tool

- **SMV, SPIN**
- **Used SMV**

**SMV Model Checking Tool**

- **SMV program**
  - **Modules**
    - `MODULE proc(state0, state1, turn, turn0)`
    - Defined proc as a module with four formal parameter
  - **Variables declared in Module**
    - Type: boolean, enumeration type, integer subrange
  - **Example VAR state0: (noncritical, trying, critical, ready);**
  - **Structural hierarchy**
    - Module may contain instances of other module

**SMV program (Cont.)**

- Contains main with no formal parameters
- main root of model hierarchy

**SMV program (Cont.)**

- Values of Variables in each state defined using init and next
- Value of variable in next state: function of value of value of variables in current state
- Choice is made non deterministically
- Example: Init(state0) := noncritical
  Next(state0) := case
  (state0 = noncritical) : {trying, noncritical}
  (state0 = trying) & ((state1 = noncritical) | (state1 = trying)) : ready

**Example of SMV Program**

```plaintext
MODULE proc(state0, state1, turn, turn0)
ASSIGN init(state0) := noncritical;
next(state0) := case
  (state0 = noncritical) : (trying, noncritical);
  (state0 = trying) & ((state1 = noncritical) | (state1 = trying) | (state1 = ready)) : ready;
  (state0 = ready) : critical;
  (state0 = trying) & (state1 = trying) & (turn = turn0) : critical;
  (state0 = critical) : (critical, noncritical);
```

**Temporal Logic Formula**

- **Ensures Mutual Exclusion**
  - Mutual exclusion is specified by the following temporal logic formulas:
    - `SPEC AG((s0 = critical) -> !(s1 = critical))`
    - `SPEC AG((s1 = critical) -> !(s0 = critical))`
  - `AG p` means that in all possible execution sequences (specified by the A part), it is globally true (the G part) that `p` holds. In other words, `p` is invariant.
  - In this case we are saying that once a process is in the critical region, the other process cannot be in its critical region.
Temporal Logic Formula

- **Concept of Invariant**
  - SPEC AG((s0 = trying) → AF(s0 = critical))
  - SPEC AG((s1 = trying) → AF(s1 = critical))

Another useful property is expressed by the formulas above. They state that an invariant of the model is the fact that if a process is in the trying region, then in all possible execution sequences, at some point in the future (indicated by the $F$ part), it will be in the critical region.

SMV Model Checking Tool

Advantages Model Checking

- **Advantages**
  - Communication System of NetKuang expensive to deploy
  - Size of state space limited for search engines
  - Model Checking can look for different possibilities
  - Temporal logics implement security policies efficiently and economically

Description Of The Model

- **Four Elements**
  - Hosts
  - Connectivity
  - Attacker Point of View
  - Exploits

Hosts

- **Set of Vulnerabilities**
  - Observable System attribute which may be a prerequisite for an exploit
  - Security problems
    - Example: Running an outdated version of sendmail
    - Configuration Information about the host
      - OS type and version, type of Authentication, max length of passwords and network services

Hosts (Cont.)

- **Current Access Level of Attacker to execute programs on Host**
  - Default: User rights by current access level
  - none, root

Connectivity

- **host’s ability to communicate with other hosts in the model**
  - Look for filters
  - Do not change during analysis
  - Changes in filtering accounted by attacker point of view
**Attacker Point of View**
- Host used by attacker for attack
- After a host is compromised attacker launch exploits further
- May circumvent network filters

**Exploits**
- Defined by
  - Set of vulnerabilities
  - Source access level
  - Target access level
  - Connectivity
  - Affects changes to security of hosts to make model dynamic
  - Direct relation with quality of analysis

**Description Of The Model**

**Initialization of the Model**
- **Four Parts**
  - Exploit description
  - Host Initialization
  - Connectivity description
  - Failure definition

**Exploit description**
- If \((\text{Boolean statement} = \text{True}) \& \& (\text{Connectivity-host} = 1)\) then exploit succeed
  - Host updated according to the exploit
  - Example: Additional vulnerabilities added to host
  - Change to attacker’s current access level on the host

**Host Initialization**
- **Host initialization**
  - Review configuration of each host and check for vulnerabilities in the host.
    - Can use COPS, ISS
    - Tool to be customized to look for prerequisite vulnerabilities
  - Initialize Access level for each host
    - Advantage: Can account for both outsiders and insiders

**Example:**
<table>
<thead>
<tr>
<th>Pre-requisite</th>
<th>Source</th>
<th>Target</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Versions Up to 1.0.4</td>
<td>ANY</td>
<td>ANY</td>
<td>Access level changed to httpd</td>
</tr>
</tbody>
</table>
Host initialization

**Example**

<table>
<thead>
<tr>
<th>Vulnerabilities</th>
<th>Current Access Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solaris Version 2.5.1</td>
<td>None</td>
</tr>
<tr>
<td>Apache Version 1.04</td>
<td></td>
</tr>
<tr>
<td>Count.cgi</td>
<td></td>
</tr>
<tr>
<td>File.cgi</td>
<td></td>
</tr>
<tr>
<td>Telnetd</td>
<td></td>
</tr>
<tr>
<td>Pipp</td>
<td></td>
</tr>
<tr>
<td>displaydb</td>
<td></td>
</tr>
</tbody>
</table>

Connectivity description

**Connectivity Matrix**

- Can use port numbers to enrich description

<table>
<thead>
<tr>
<th></th>
<th>Private File Server</th>
<th>Public Web Server</th>
<th>Border Router</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacker</td>
<td>N/A</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Border Router</td>
<td>Yes</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>Public Web Server</td>
<td>Yes</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>Private File Server</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Failure definition

- Invariant Statements - Should be true in every state
- Example: AG PrivateFileServer.Access = None
- If not then report failure

Analyses Method

- Keeping view of
  - Attacker access
  - Prerequisite Host vulnerabilities for an exploit
- Model can change
  - State based on rules defined for exploit
  - Result in additional vulnerabilities added to target
  - May update attacker’s access level on host

Analyses Method (Cont.)

- With change of state of model
  - Security of the network reduces
- Stopping criteria
  - Either invariant statement turn out to be violated
  - Or no more exploits can be employed
Counterexamples

- Represent series of exploits to be run
- Till invariant has been violated
  - Example: AG \( \neg \text{host} \text{.access} = \text{root} \)
- Represent an attacker's scenario

**Example**

**Border Filtering Rules**

<table>
<thead>
<tr>
<th>Source Address</th>
<th>Destination Address</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any</td>
<td>192.168.1.4</td>
<td>Deny</td>
</tr>
<tr>
<td>192.168.1.0/24</td>
<td>Any</td>
<td>Allow</td>
</tr>
<tr>
<td>Any</td>
<td>Any</td>
<td>Deny</td>
</tr>
</tbody>
</table>

**Hosts**

- Module machine
- Var
  - access : \{none, user, root\}
  - exploit : array 1..6 of boolean
  - hostid : \{1, 2, 3, 4\}
  - vulnerability : array 1..15 of boolean

**Initialization:** each variable in host given specific initial value
- Init (exploit[1]) := 0;
- Init (exploit[2]) := 0;
- Init (exploit[3]) := 0;
- Init (exploit[4]) := 0;
- Init (exploit[5]) := 0;
- Init (exploit[6]) := 0;

**Vulnerabilities**

- Init ((PublicWebServer.vulnerability[1]) := 1;
- Apache/1.04
- Init ((PublicWebServer.vulnerability[2]) := 0;
- - home directories exported rw (ALL)
- Init ((PublicWebServer.vulnerability[3]) := 0;
- - ftp
- Init ((PublicWebServer.vulnerability[4]) := 0;
- - nfid
- Init ((PublicWebServer.vulnerability[5]) := 1;
- - no shadow file
Hosts (Cont.)
- Access
  - For an external attack
    * Init(PublicWebServer.access) := none;

Exploits (Cont.)
- Attack module
  - Value of "a" varies non deterministically from 1 to total number of exploits. To check if an exploit has been run more than once.

Exploits
- Attack module
  - For an external attack
    * Init(PublicWebServer.access) := none;

Connectivity Matrix
- Init(connect[1][1]) := 1; next(Connect[1][1]) := 1;
  - attacker to attacker
- Init(Connect[1][2]) := 1; next(Connect[1][2]) := 1;
  - attacker to border router
- Init(Connect[1][3]) := 1; next(Connect[1][3]) := 1;
  - attacker to PublicWebServer
- Init(Connect[1][4]) := 1; next(Connect[1][4]) := 1;
  - attacker to PrivateFileServer

Exploits
- Attack module
  - Example: Phf vulnerability exploit
    next(m.exploit[4]) := - PHF.cgi
    case
      - current exploit number
      a := 4
    - check for connectivity
      src = 1 & m.hostid = 1 & conn[1][1] & conn[1][2] & conn[1][3] & conn[1][4]
      - check for required prerequisites
        m.vulnerability[1] & m.vulnerability[6]

Exploits (Cont.)
- Result module
  next(m.vulnerability[7]) :=
    case
      m.exploit[3] := 1;
    - capture password hashes
      1: m.vulnerability[7];
    esac
  - Setting Access level
    next(m.access) :=
    case
      m.exploit[5] := root;
    esac
Counter Examples

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Exploit</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hacker</td>
<td>Public Web Server</td>
<td>Phf</td>
<td>User access on Public Web Server</td>
</tr>
<tr>
<td>Hacker</td>
<td>Private File Server</td>
<td>Capture pwd hashes</td>
<td>Hacker knows Private File Server’s root password is known to hacker</td>
</tr>
<tr>
<td>Hacker</td>
<td>Public Web Server</td>
<td>Remote Force Passwords</td>
<td>Hacker knows Public Web Server’s root password is known to hacker</td>
</tr>
<tr>
<td>Hacker</td>
<td>Public Web Server</td>
<td>Shell login as root</td>
<td>Hacker’s access level on Public Web Server changed to root</td>
</tr>
<tr>
<td>Public Web Server</td>
<td>Private File Server</td>
<td>Shell login as root</td>
<td>Hacker’s access level on Private File Server changed to root</td>
</tr>
</tbody>
</table>

Conclusions

- Model multiple attack scenarios ??

Mutating Network Models to Generate Network Security Test Cases

Ronald W. Ritchey

Mutating Model

- Mutation analysis to generate test cases for network security
- Define mutant operators
- Each version represent a mutant of original program

Defining Mutant Operators

- Purpose: To make MODEL less secure to create different real world scenarios
- Source: Exploit prerequisite
- Operators
  - Adding vulnerabilities
  - Increasing access levels
  - Adding connectivity

Adding connectivity

- capture firewall’s changes in its rule set
  - Example: to allow more traffic
- Analyze demonstrate level of access an attacker can gain by change of policy
  - Example: Allow attacker direct access to private file server
Increase Access Level

To answer what if an insider attack?

Add Vulnerability

To capture configuration changes
- Example: Adding software, changing permissions, modifying settings etc.
- Feed only feasible vulnerabilities
- Constraint: can see only known vulnerabilities
- Can't account for an unknown one

Coverage Criterion

Number of mutant operators that can be applied together to produce a counterexample
- Coverage level one then
  - Account for any single configuration changes
- Coverage level two then
  - Account for two configuration changes
- Advantage: The higher coverage level more secure will be the network

Running the analysis

Coverage Criterion

Security Recommendations

<table>
<thead>
<tr>
<th>Number</th>
<th>Mutation</th>
<th>Score</th>
<th>1st English</th>
<th>Security Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Add Connectivity between File Server and Public Web Server</td>
<td>3</td>
<td>Add RFD trust to Public Web Server</td>
<td>Eliminate RFD trust to Public Web Server</td>
</tr>
<tr>
<td>2</td>
<td>RDP access to the Public Web Server</td>
<td>2</td>
<td>Eliminate RDP trust to Public Web Server</td>
<td>Add RDP trust to Public Web Server</td>
</tr>
<tr>
<td>3</td>
<td>Anonymous login on Public Web Server</td>
<td>1</td>
<td>Eliminate RDP trust to Public Web Server</td>
<td>Add RDP trust to Public Web Server</td>
</tr>
</tbody>
</table>

Security Recommendations (Cont.)

<table>
<thead>
<tr>
<th>Number</th>
<th>Mutation</th>
<th>Score</th>
<th>1st English</th>
<th>Security Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Use Password Hash on Public Web Server</td>
<td>6</td>
<td>Use strong authentication on Public Web Server</td>
<td>Eliminate RDP trust to Public Web Server</td>
</tr>
<tr>
<td>5</td>
<td>RDP trust to Public File Server</td>
<td>5</td>
<td>Eliminate RDP trust to Public File Server</td>
<td>Add RDP trust to Public File Server</td>
</tr>
<tr>
<td>6</td>
<td>Use Password Hash on Public Web Server</td>
<td>4</td>
<td>Use strong authentication on Public Web Server</td>
<td>Eliminate RDP trust to Public Web Server</td>
</tr>
</tbody>
</table>
Discussion

- Any Question???