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 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)

Model Checking Tools

- 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 Model Checking Tool

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

SMV Model Checking Tool

SMV program (Cont.)

- Example of SMV Program

```
MODULE prc(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};
```

SMV Model Checking Tool

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.

SMV Model Checking Tool

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

Description Of The Model

Hosts (Cont.)

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

Description Of The Model

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

Description Of The Model

Attacker Point of View

- host used by attacker for attack
- After a host is compromised attacker launch exploits further
- May circumvent network filters

Description Of The Model

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

- Exploit description
  - pre-requisite Vulnerabilities
  - Source access level
  - Target access level
- Info converted into Boolean statement
- If ((Boolean statement = True) && (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

Initialization of the Model

Exploit description

- If ((Boolean statement = True) && (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

Example:

Prerequisites	Source Access Level	Target Access Level	Results
Apache Version Up to 1.0.4	ANY	ANY	Access level changed to httpd

Initialization of the Model

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

Initialization of the Model

Host initialization

- Example

Vulnerabilities	Current Access Level
Solaris Version 2.5.1 Apache Version 1.04 Count.cgi Phf.cgi Telnetd Ftpd dtappgahther	None

Initialization of the Model

Connectivity description

- Connectivity Matrix
- Can use port numbers to enrich description

Initialization of the Model

Connectivity description

- Connectivity Matrix

	attacker	Border Router	Public Web Server	Private File Server
Attacker	N/A	Yes	Yes	No
Border Router	Yes	N/A	Yes	Yes
Public Web Server	Yes		N/A	Yes
Private File Server	No	Yes	Yes	N/A

Initialization of the Model

Failure definition

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

Initialization of the Model

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 !host.access = root
  - Represent an attacker's scenario

Example

- Border Filtering Rules

Source Address	Destination Address	Action
Any	192.168.1.4	Allow
192.168.1.0/24	Not 192.168.1.4	Allow
Any	Any	Deny

Encoding the example model in SMV

- 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

Hosts (Cont.)

- 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;

Encoding the example model in SMV

Hosts (Cont.)

- Hostid
- Attacker.hostid :=1;
- Init(BorderRouter. hostid) :=2;
- Next ((BorderRouter. hostid) :=2;
- Init(PublicWebServer.hostid) :=3;
- next(PublicWebServer.hostid) :=3;
- Init(PrivateFileServer.hostid) :=4;
- next(PrivateFileServer.hostid) :=4;

Encoding the example model in SMV

Hosts (Cont.)

- Vulnerabilities
- Init (PublicWebServer.vulnerability[1]):=1;
  - Apache/1.04
- Init (PublicWebServer.vulnerability[2]):=0;
  - home directories exported rw (ALL)
- Init (PublicWebServer.vulnerability[3]):=0;
  - ftpd
- Init (PublicWebServer.vulnerability[4]):=0;
  - nfsd
- Init (PublicWebServer.vulnerability[5]):=1;
  - no shadow file

Encoding the example model in SMV

Hosts (Cont.)

- Access
- For an external attack
  - Init(PublicWebServer.access) :=none;

Encoding the example model in SMV

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]):=0;next(Connect[1][4]):=0;
  - attacker to PrivateFileServer

Encoding the example model in SMV

Exploits

- Attack module
- Result module

Encoding the example model in SMV

Exploits (Cont.)

- 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]) |src = 1 & m.hostid =2 &conn[1][2]) |
src = 1 & m.hostid =3 & conn[1][3]) |src = 1 & m.hostid =4 & conn[1][4]) |
src = 2 & m.hostid =1 & conn[2][1]) |src =2 & m.hostid =2 & conn[2][2]) |
src = 2 & m.hostid =3 & conn[2][3]) |src = 2 & m.hostid = 4 & conn[2][4]) |
src = 3 & m.hostid =1 & conn[3][1]) |src = 3 & m.hostid =2 & conn[3][2]) |
src = 3 & m.hostid =3 & conn[3][3]) |src = 3 & m.hostid =4 & conn[3][4]) |
src = 4 & m.hostid =1 & conn[4][1]) |src = 4 & m.hostid =2 & conn[4][2]) |
src = 4 & m.hostid =3 & conn[4][3]) |src = 4 & m.hostid =4 & conn[4][4]) ) &
- check for required prerequisite
m.vulnerability[1] & m.vulnerability[6]
```

Encoding the example model in SMV

Exploits (Cont.)

- Attack module

  Value of “a” varies non deterministically from 1 to total number of exploits. To check if an exploit has been not run more than once.

Encoding the example model in SMV

Exploits (Cont.)

- Result Module
- next(m.vulnerability[7]) :=   - password hashes known

```
case
m.exploit[3] :  1;
- capture password hashes
1 : m.vulnerability[7];
esac;
```

- Setting Access level

```
next(m.access):=
case
m.exploit[4] | m.exploit[6] : user;
m.exploit[5]: root;
1:   m.access;  esac;
```

Encoding the example model in SMV

Counter Examples

Source	Target	Exploit	Result
Hacker	Public Web Server	Phf	User access on Public Server
Hacker	Public Web Server	Capture pwd hashes	Public Web Server's password hashes known to hacker
Hacker	Public Web Server	Brute Force Passwords	Hacker knows Public Web Server's root password
Hacker	Public Web Server	Shell login as root	Hacker's access level on Public Web Server changed to root
Public Web Server	Private File Server	Shell login as root	Hacker's access level on Private File Server changed to root

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?

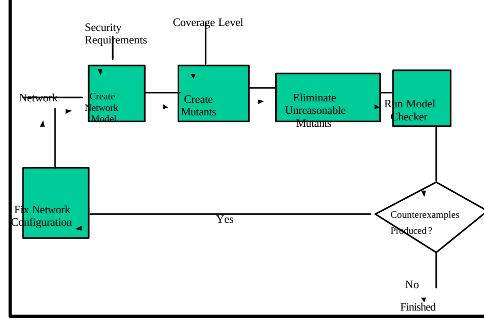
Add Vulnerability

- ✍ To capture configuration changes
  - Example: Adding software, changing permissions, modifying settings etc.
  - Feed only feasible vulnerabilities
  - Constraint: can see only known vulnerabilities
  - Cant 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



Security Recommendations

Number	Mutant	1^{st} Exploit	Security Recommendation
1	Add Connectivity from Attacker to Private File Server	Add BSD trust from Attacker to Private File Server	Eliminate BSD daemons on Private File Server
2	Add PHF program to Public Web Server	Use PHF to gain user access to Public Web Server	Verify PHF not on Public Web Server
3	Password Hashes known on Public Web Server	Brute Force Passwords on Public Web Server	Use strong authentication on Public Web Server

Security Recommendations (Cont.)

Number	Mutant	1^{st} Exploit	Security Recommendation
4	Root Password known to Public Web Server	Telnet to Public Web Server	Use strong authentication on Public Web Server
5	BSD trust between Attacker and Public Server	rlogin to Public Web Server	Eliminate BSD daemons on Public Web Servers
6	User Passwords known on Public Web Server	Telnet to Public Web Server	Use strong authentication on Public Web Server

Discussion

- Any Question???