Announcements

- Project progress report due Thursday at 11:59pm (by email)
- Guest lectures
  - Thursday: David Brumley
    - Application partitioning
    - Timing attacks
  - Tuesday: Liam O'Brien
    - Re-engineering

Tools for Generating and Analyzing Attack Graphs

Paper by Oleg Sheyner
and Jeannette Wing

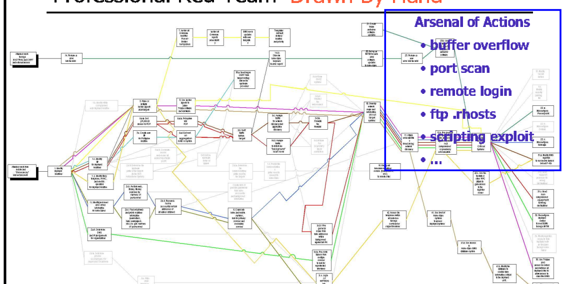
Slides adapted from a presentation by Jeannette Wing
Used by permission

17-654/17-765
Analysis of Software Artifacts
Jonathan Aldrich

Attack Graphs

- Analyzing the security of a network
  - Heterogeneous hardware and software
  - Complex connectivity
  - Difficult to ensure complete lack of security holes
- Defense in depth
  - Multiple layers
    - Multiple firewalls
    - Authentication
    - Limited privilege
  - Achieving root access on machine X may require multiple steps
    - Get inside firewall
    - Scan network for vulnerabilities
    - Get user access to machine
    - Get root access to machine
- Question: how does security of whole system depend on parts?

Example of Attack Graph Developed by a Professional Red Team Drawn By Hand

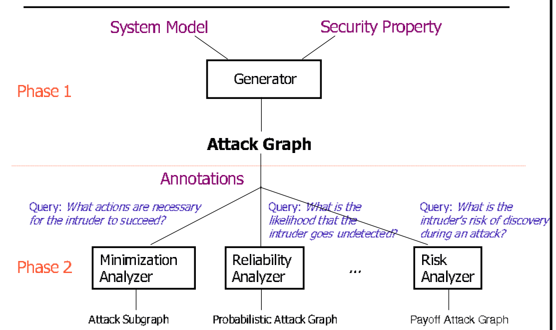


Sandia Red Team "White Board" attack graph from DARPA CC20008 Information battle space preparation experiment

Problem Statement

- Problem: Generating attack graphs by hand is tedious, error-prone, and impractical for large systems.
- Our Goal: Automate the generation and analysis of attack graphs.
  - Generation
    - Must be fast and completely automatic
    - Must handle large, realistic examples
    - Should guarantee properties of attack graphs
  - Analysis
    - Must enable security analysis by system administrators
    - Should support incremental, partial specification

Overview of Our Method



Why Model Checking?

- Pragmatic reasons
  - Off-the-shelf technology
  - Major verification success story
- Technical reasons
  - Fast, automatic
  - Large state spaces
  - Handles safety and liveness properties
  - *Generates counterexamples*

Counterexample = Attack

$\Phi \equiv$ AG p

single counterexample = violation of Φ
= path by which intruder succeeds
= attack

For example,

$\Phi \equiv$ AG (intruder does not have admin access to host H)

Hence, an attack (violation of Φ) is an example of how the intruder can gain unauthorized access to H.

Definition of Attack Graph

- Given
  - a finite state model, M, of network
  - a security property Φ
- An attack is an execution of M that violates Φ.
- An attack graph is a set of attacks of M.

Properties of Attack Graphs

- Sound
  - An attack generated violates Φ.
- Exhaustive
  - All possible attacks are represented in G.
- Succinct
  - Only relevant states are contained in G.
  - Only relevant transitions are contained in G.

We developed two algorithms that satisfy these properties.

Explicit-State Attack Graph Generation Algorithm

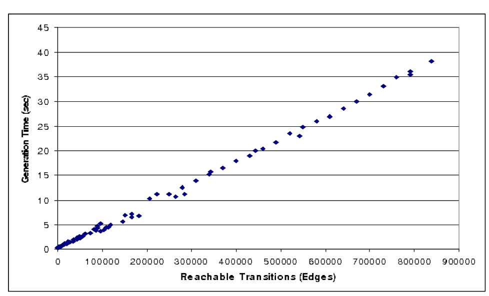
Inputs
- M
- Φ = LTL property (safety or liveness)

$\Box$*(request* $\Rightarrow$ $\Diamond$*(response))*

Output
- Attack graph G s.t. L(G) = L(M)\L(Φ)

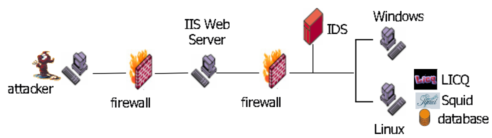
Algorithm
1. Interpret network model M and security property Φ as Buchi automata [Gerth et al.95].
   - M and Φ induce languages L(M) and L(Φ).
2. Compute intersection M $\cap$ ~Φ of Buchi automata.
   - L(M $\cap$ ~Φ) = L(M)\L(Φ) = executions of M that violate Φ.
3. Derive G from strongly connected components of intersection automaton [Tarjan72].

Performance (Explicit-State)



Linear coefficient 1.12 x 10^{-4} Linear Regression **R^2** = 0.9967

An Illustrative Example



Action Arsenal		Always Detected
IIS buffer overflow:	remotely get root	✗
Squid portscan:	port scan	✗
LICQ remote-to-user:	gain user privileges remotely	✓
scripting exploit:	gain user privileges remotely	✗
local buffer overflow:	locally get root	✗

Modeling a Network and Intruder

- Set of hosts H
  - running services
  - CVE vulnerabilities
  - trust relationships
  - misc. configuration
- Set of networks N
  - each network $n \in N$ is a subset of H
  - connectivity: $P \subset N \times N$ models firewalls, packet filter rules, physical links
- Intrusion detection systems
  - placement: $P \subset N \times N$
  - detectability per action
- Intruder
  - store of knowledge
  - privileges on each host
- Set of actions A
  - preconditions
  - postconditions

Modeling Attacks

IIS Buffer Overflow. This remote-to-root action immediately gives a remote user a root shell on the target machine.

action IIS-buffer-overflow **is**	
intruder preconditions	
$plvl(S) \geq$ user	*User-level privileges on host S*
$plvl(T) <$ root	*No root-level privileges on host T*
network preconditions	
$\mathrm{w3svc}_T$	*Host T is running vulnerable IIS server*
$R(S, T, 80)$	*Host T is reachable from S on port 80*
intruder effects	
$plvl(T) :=$ root	*Root-level privileges on host T*
network effects	
$\neg\mathrm{w3svc}_T$	*Host T is not running IIS*
end	

Modeling Attacks

Squid Port Scan. The *Squid* port scan action uses a misconfigured *Squid* web proxy to conduct a port scan of neighboring machines and report the results to the intruder.

action squid-port-scan **is**	
intruder preconditions	
$plvl(S) =$ user	*User-level privileges on host S*
$\neg$scan	*We have not yet performed a port scan*
network preconditions	
squid_T	*Host T is running vulnerable* Squid *proxy*
$R(S, T, 80)$	*Host T is reachable from S on port 80*
intruder effects	
scan	*We have performed a port scan on the network*
network effects	
$\emptyset$	*No changes to the network component*
end	

Modeling Attacks

LICQ Remote to User. This remote-to-user action immediately gives a remote user a user shell on the target machine. The action rule assumes that a port scan has been performed previously, modeling the fact that such actions typically become apparent to the intruder only after a scan reveals the possibility of exploiting software listening on lesser-known ports.

action LICQ-remote-to-user **is**	
intruder preconditions	
$plvl(S) \geq$ user	*User-level privileges on host S*
$plvl(T) =$ none	*No user-level privileges on host T*
scan	*We have performed a port scan on the network*
network preconditions	
licq_T	*Host T is running vulnerable LICQ software*
$R(S, T, 5190)$	*Host T is reachable from S on port 5190*
intruder effects	
$plvl(T) :=$ user	*User-level privileges on host T*
network effects	
$\emptyset$	*No changes to the network component*
end	

Modeling Attacks

Scripting Action. This remote-to-user action immediately gives a remote user a user shell on the target machine. The action rule does not model the social engineering required to get a user to download a specially-created Web page.

action client-scripting **is**	
intruder preconditions	
$plvl(S) \geq$ user	*User-level privileges on host S*
$plvl(T) =$ none	*No user-level privileges on host T*
network preconditions	
$\mathrm{scripting}_T$	*HTML scripting is enabled on host T*
$R(T, S, 80)$	*Host S is reachable from T on port 80*
intruder effects	
$plvl(T) :=$ user	*User-level privileges on host T*
network effects	
$\emptyset$	*No changes to the network component*
end	

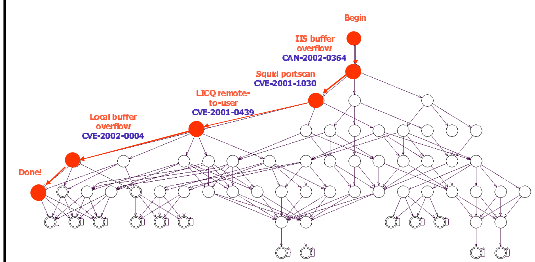
Modeling Attacks

Local Buffer Overflow. If the intruder has acquired a user shell on the target machine, this action exploits a buffer overflow vulnerability on a *setuid root* file (in this case, the *at* executable) to gain root access.

action local-setuid-buffer-overflow **is**
  intruder preconditions
    $plvl(T) = \text{user}$ — *User-level privileges on host T*
  network preconditions
    vul-at_T — *There is a vulnerable* at *executable*
  intruder effects
    $plvl(T) := \text{root}$ — *Root-level privileges on host T*
  network effects
    $\emptyset$ — *No changes to the network component*
end

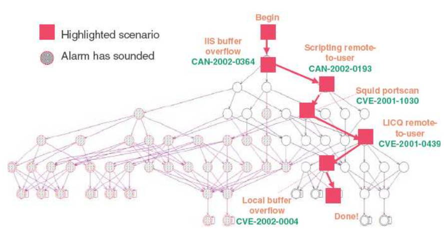
Example Attack Graph

Φ = G (intruder.privilege(Linux) < root)

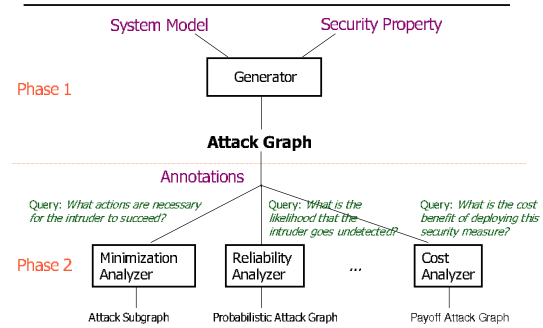


Example Attack Graph: avoiding the IDS

Φ = G (intruder.privilege(Linux) < root)



Overview of Our Method



Minimization Analysis

Scenario: The system analyst must decide
- among several different firewall configurations, or
- among several vulnerabilities to patch, or
- among several intrusion detection systems to set up,

each of which prevents different subsets of actions.
What should he do?

Problem Question (Minimum Critical Set of Actions): What is a minimum set of actions that must be prevented to guarantee the intruder cannot achieve his goal?

Solution (Sketch):
1. Reduce MCSA to Minimum Hitting Set (MHS) Problem [JSW02].
2. Reduce MHS to Minimum Set Covering (MSC) Problem [ADG80].
3. Use textbook Greedy Approximation Algorithm to approximate solution [CLR85].

Minimum Critical Set of Actions

A = the set of actions available to the intruder

Def 1: A set of actions C is critical if the intruder cannot achieve his goal using only actions in A \ C.

Def 2: A critical set of actions C is minimum if there is no critical action set of smaller size.

Def 3: A set of actions A′ $\subseteq$ A is realizable if the intruder can achieve his goal using only actions in A′.

Minimum Critical Set of Actions (MCSA):
Given a set of actions A and an attack graph G, find a minimum critical action subset C $\subseteq$ A.

Finding a minimum set: NP-complete

Reduction to Minimum Hitting Set Problem

Minimum Hitting Set (MHS):

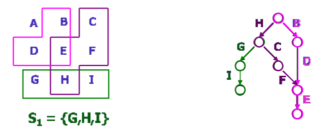
Given a collection C of subsets of a finite set S, find a minimum subset $S' \subseteq S$ such that each subset in C contains at least one element from S′.

MCSA: Collection of realizable sets of actions ⟺ MHS: Collection of subsets C

MCSA and MHS are polynomially-equivalent.

[JSW02b] Jha, Sheyner, Wing, "Two Formal Analyses of Attack Graphs," Computer Security Foundations Workshop, Nova Scotia, June 2002.

Sketch of Reduction from MCSA to MHS



$S_1 = \{G,H,I\}$

$S_2 = \{C,E,F,H\}$

$S_3 = \{B,D,E\}$

E.g., **S′ = {H, D}**

Reduction of MHS to Minimum Set Covering

Minimum Set-Covering (MSC):

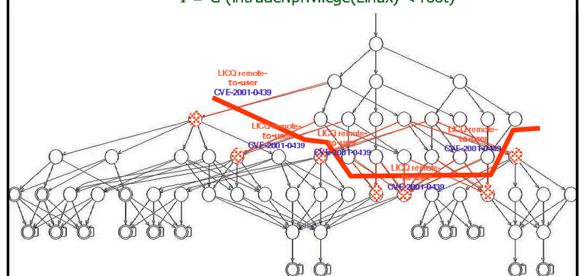
Given a collection C of subsets of a finite set S that covers S, find a minimum sub-collection $C' \subseteq C$ that covers S.

MHS and MSC are polynomially-equivalent [ADP80].

Use textbook Greedy Approximation Algorithm for MSC [CLR85, p. 975.]

LICQ Coverage

Φ = G (intruder.privilege(Linux) < root)



Other Minimization Analyses [S04, JSW02b]

Scenario: The system analyst has a set of measures, M, each of which prohibits a subset of actions.

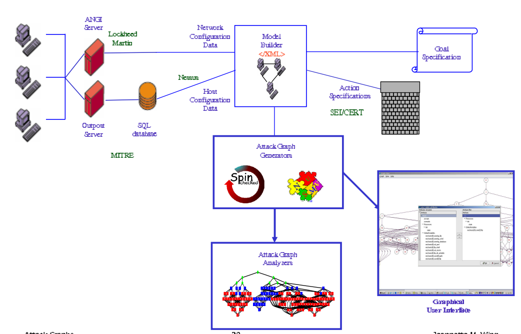
E.g., M = {packet filter firewall, application firewall, smart cards, one-time passwords, authentication policy servers, VPNs, anti-virus software, email filters, database encryption, host-based IDS, net-based IDS, network monitors, auditing, key stroke replicator, log analysis, forensic software, hardened O/S}

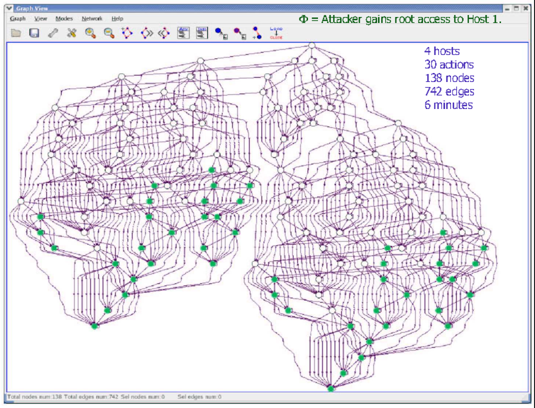
Problem Question: What is a smallest subset of measures he can deploy to make the system safe? [S04]

Solution Approach: Greedy algorithm with provable bounds. General case is NP-complete (slightly more complex than minimum cover problem).

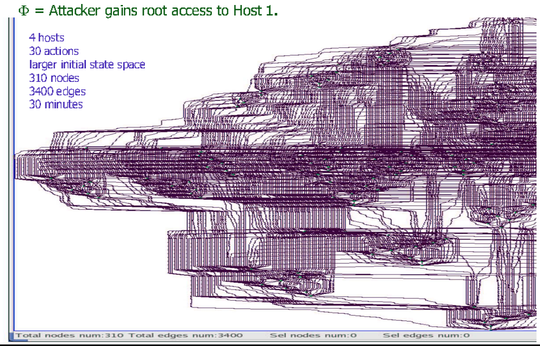
Status of Tool Suite

Thanks to Oleg Sheyner,, Roman V. Lototski, Alexey Roschyna, Arvind Kannan, and Meera Sridhar





A Graph Larger than Fits on Your Screen



XML Specification of a Host

```
<host name="lin" ip="192.168.0.4" network="internal">
  <services>
    <Squid/>
    <LICQ/>
    <database/>
  </services>
  <connectivity>
    <remote id="ferrari">
        <W3SVC/> </remote>
    <remote id="emilla">
        <ftp/> <sshd/> </remote>
  </connectivity>
  <cve>
    <CVE 2002 0004/>
    <CVE 2001 1030/>
    <CVE 2001 0439/>
  </cve>
</host>
```

```
<host name="lin" ip=|Outpost|>
  <services source=|Nessus|>
  <connectivity source=|ANGI|>
  <cve source=|Outpost|>
</host>
```


Current Work

- Input to graph generation
  - Building a library of action specifications
    - To describe majority of CERT advisories, MSR security bulletins, Symantec, …
    - Starting point: CERT database of 100+ rule-based specs
  - Goal: Discover new attacks
- More experimentation and analyses
  - Run tools over different security properties and system models
    - Goal: Push on limits of state-space explosion problem.
  - Dynamic analysis
    - Goal: Adapt to on-going attacks.
- Scenario graphs
  - Application to other domains, e.g., test-case generation, embedded systems