Information Trust INSTITUTE

Probabilistic Validation of Computer System Security

William H. Sanders
University of Illinois
(Joint work with DPASA Project Team)

Everyone says it is important, few approaches exist …

- Security metrics were an important problem in the 2005 INFOSEC Research Council Hard Problems List
- New security metrics that are linked to the business were ranked first among six key security imperatives developed by over twenty Fortune 500 firms
- New regulatory requirements of Sarbanes-Oxley and the Basel II Accord have created more urgency for metrics that integrate security risk with overall business risk
- Almost every critical infrastructure roadmap lists security metrics as a critical challenge
- The list goes on …

Security Validation Truths …

- Security is no longer absolute
- Trustworthy computer systems/networks must operated through attacks, providing proper service in spite of possible partially successful attacks
- Intrusion tolerance claims to provide this ability
- If security is not absolute, quantification of the "amount" of security that a particular approach provides is essential
- Quantification can be useful in:
  - A *relative* sense, to choose amount alternate design alternatives
  - In an *absolute* sense, to provide guarantees to users

Existing Security Validation Approaches

- Most traditional approaches to security validation have focused on and specifying procedures that should be followed during the design of a system (e.g., the Security Evaluation Criteria [DOD85, ISO99]).
- When quantitative methods have been used, they have typically either been based on:
  - formal methods (e.g., [Lan81]), aiming to prove that certain security properties hold given a specified set of assumptions, or
  - been quite informal, using a team of experts (often called a "red team," e.g. [Low01]) to try to compromise a system.

Problems with Existing Approaches

- Process Guidelines can improve security, but provide NO quantification of the amount of security that has been obtained
- Formal methods aim either to prove absolute security (not usually possible), or find problems (useful, but NO quantification.
- Red Teams, can find problems (useful), but again, no quantification (sample size too small).
- Most existing metrics are lagging indicators of performance (and hence not predictive!)
- Probabilistic Methods can provide predictive quantification, but their application to security/ survivability is challenging as well.

ITI

Security Quantification Challenges

- How can the behavior of attackers be quantified?
  - How accurately does this need to be done?
  - At what level of detail?
- How should security/survivability measures be specified?
  - Are new measures needed?
- If relative measures are desired, can they be shown to be robust across a wide variety of situations?
  - Robustness is key to good design
- How accurately can absolute measures be estimated?
- Can quantification aid in security testing?
  - Knowing where to focus testing is key
- Can a notion of "coverage" be developed?
  - If so, testing can produce quantitative results

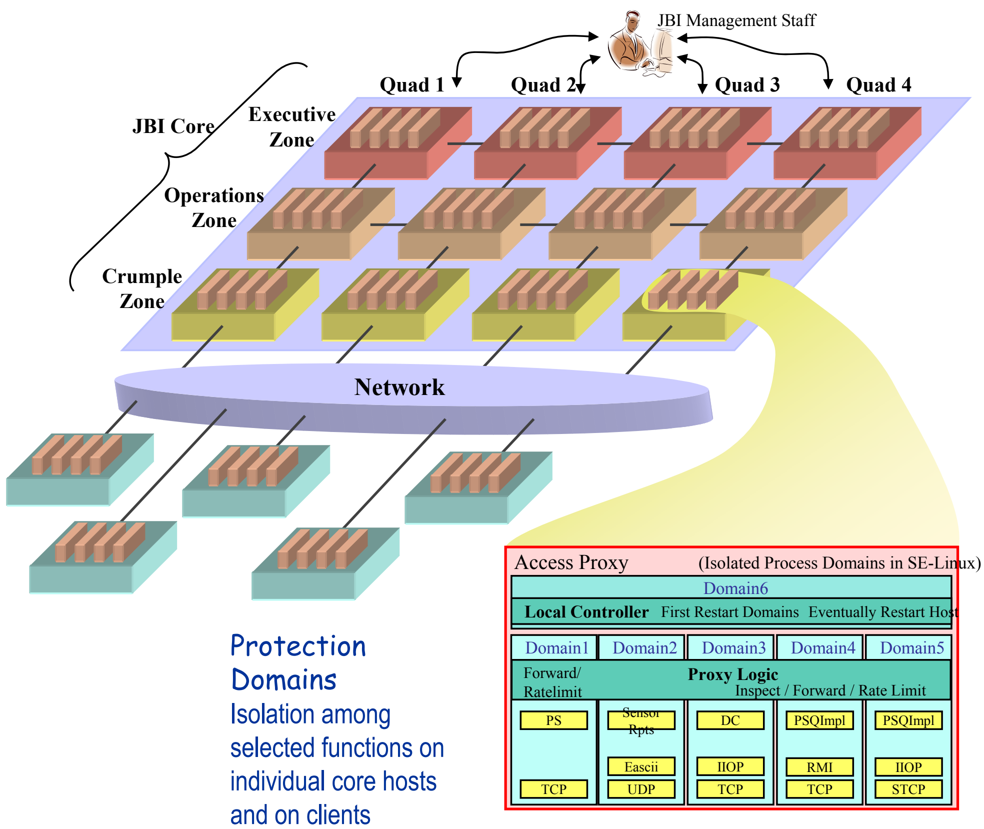
ITI

Example Probabilistic Security Validation Study

- Evaluation of DPASA-DV Project design
  - Designing Protection and Adaptation into a Survivability Architecture: Demonstration and Validation
  - USA DARPA Project, 2.5 years; 11 Million $, ~25 people on project team.
- Design of a "Joint Battlespace Infosphere"
  - Publish, Subscribe and Query features (PSQ)
  - Ability to fulfill its mission in the presence of attacks, failures, or accidents
- Goal was to design AND validate survivability of system while operating under intense attack

ITI

JBI Design Overview



Protection Domains
Isolation among selected functions on individual core hosts and on clients

ITI

Survivability/Security Validation Goal

- Phase 1: Provide convincing evidence that the design, when implemented, will provide satisfactory mission support under real use scenarios and in the face of cyber-attacks.
  - This assurance case is supported by:
    - Rigorous logical arguments
    - Experimental evaluation
    - A detailed executable model of the design
- Phase 2: Use models to guide testing of implementation in increase security test effectivness
  - Test system aspects that are most important to overall system security

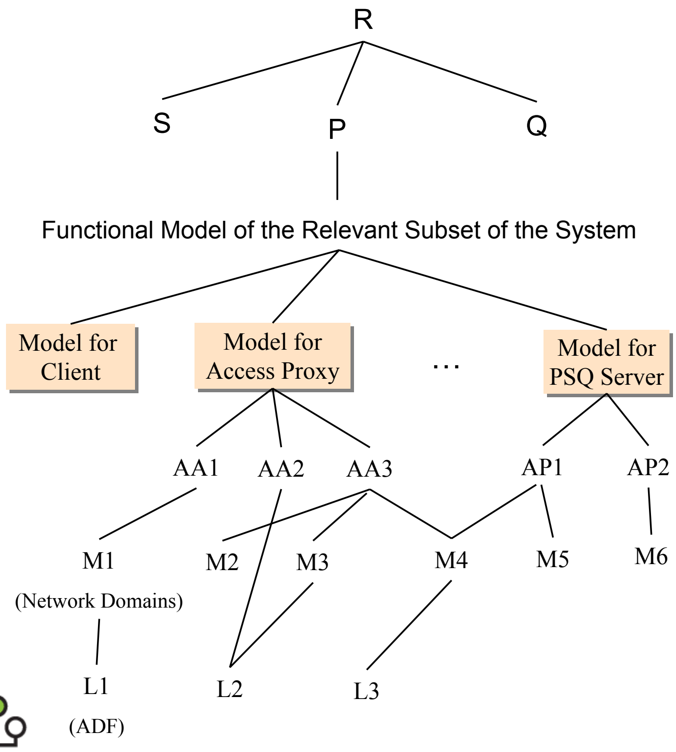
ITI

System Requirement: Design, Implement, and Validate a Publish and Subscribe Mechanism that …

- Provides 100% of critical functionality when under sustained attack by a "Class-A" red team with 3 months of planning
- Detects 95% of large scale attacks within 10 mins. of attack initiation and 99% of attacks within 4 hours with less than 1% false alarm rate
- Displays meaningful attack state alarms. Prevent 95% of attacks from achieving attacker objectives for 12 hours
- Reduces low-level alerts by a factor of 1000 and display meaningful attack state alarms.
- Shows survivability versus cost/performance trade-offs

ITI

Phase 1: Integrated Survivability Validation Procedure



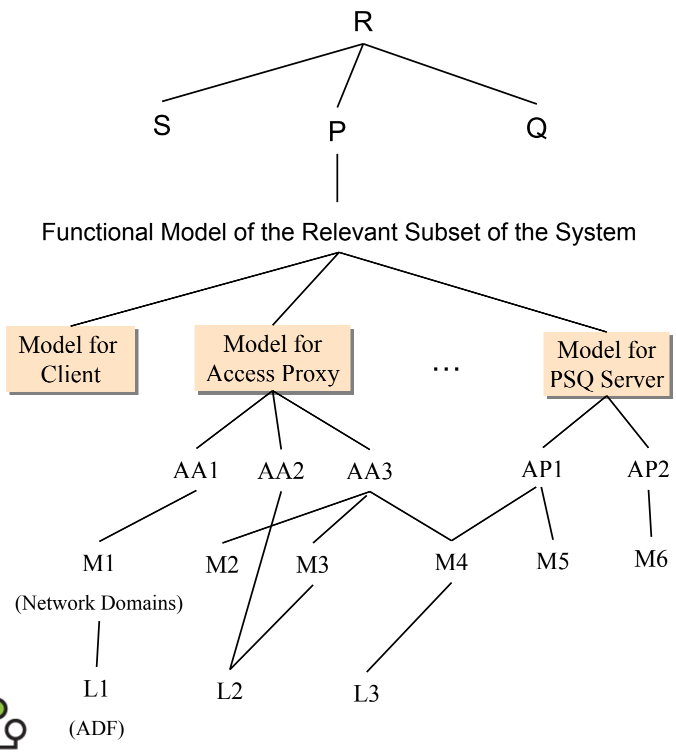
Requirement Decomposition

Functional Model of the System (Probabilistic or Logical)

Assumptions

Supporting Logical Arguments and Experimentation

Integrated Survivability Validation Procedure



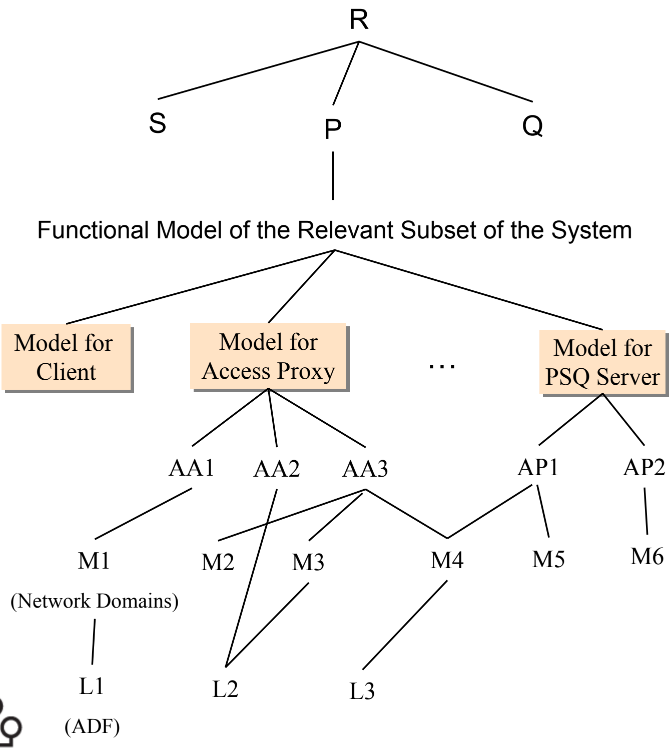
Steps

1. A precise statement of the requirements

2. High-level functional model description:
   a) Data and alerts flows for the processes related to the requirements,
   b) Assumed attacks and attack effects [Threat/vulner-ability analysis; whiteboarding]

Integrated Survivability Validation Procedure

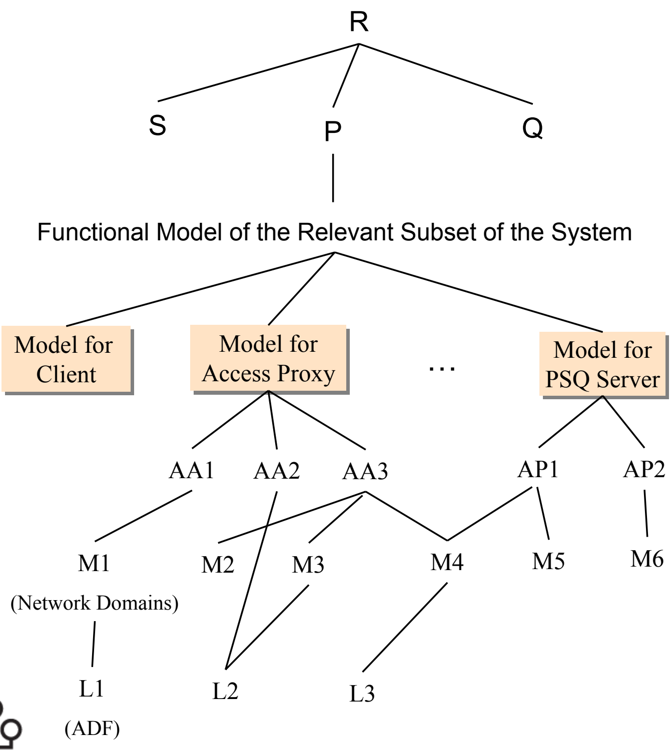
Steps



3. Detailed descriptions of model component behaviors representing 2a and 2b, along with statements of underlying assumptions made for each component. [Probabilistic modeling or logical argumentation, depending on requirement]

Integrated Survivability Validation Procedure

Steps



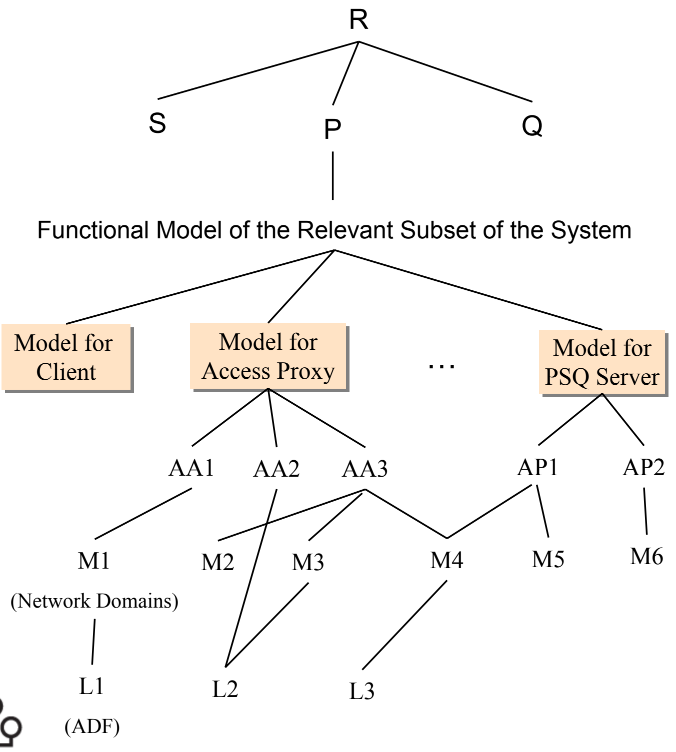
4. Construct executable functional model [Probabilistic modeling, if model constructed in 3 is probabilistic]

In Parallel

5. a) Verification of the modeling assumptions of Step 3 [Logical argumentation] and, b) where possible, justification of model parameter values chosen in Step 4. [Experimentation]

Integrated Survivability Validation Procedure

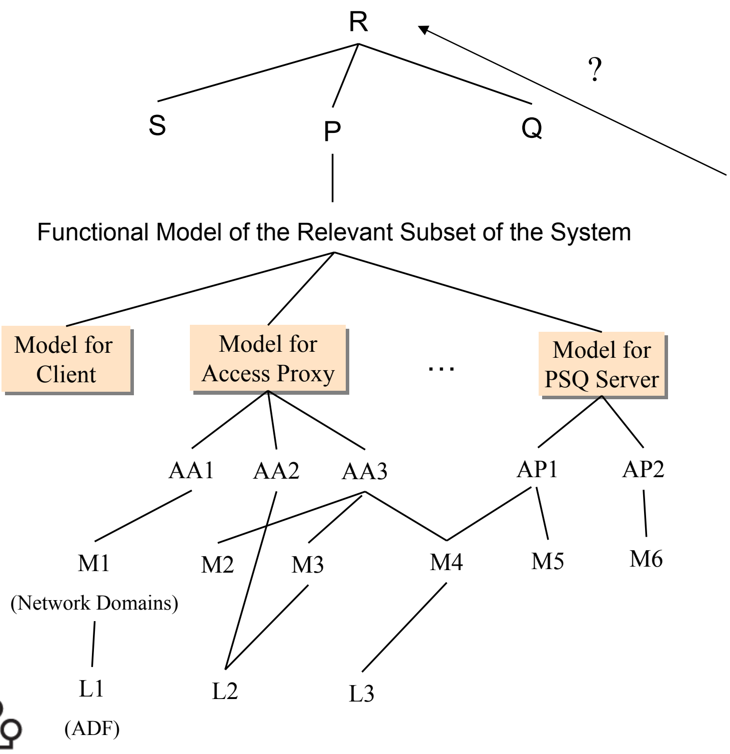
Steps



6. Run the executable model for the measures that correspond to the requirements of Step 1. [Probabilistic modeling]

Integrated Survivability Validation Procedure

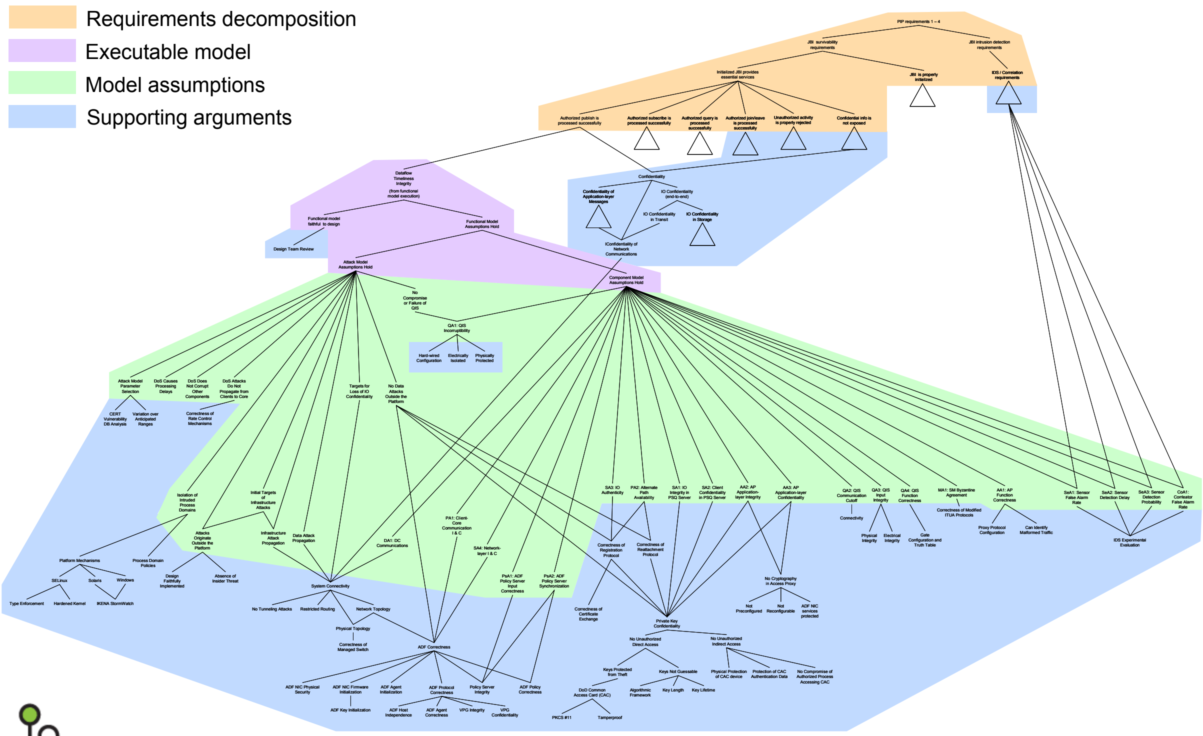
Steps



7. Comparison of results obtained in Step 6, noting in particular the configurations and parameter values for which the requirements of Step 1 are satisfied.

Note that if the requirement being addressed is not quantitative, steps 4 and 6 are skipped.

Argument Graph for the Phase 1 Design



Attack Model Description

- Consider **effects** of attacks, not attacks themselves
- Attack propagation
  - MTTD: mean time to **discovery** of a vulnerability
  - MTTE: mean time to **exploitation** of a vulnerability
- 3 types of vulnerabilities:
  - **Infrastructure-Level Vulnerabilities** → attacks in depth
    - OS vulnerability
    - Non-JBI-specific application-level vulnerability
    - p_{common} : common-mode failure
  - **Data-Level Vulnerabilities** → attacks in breadth
    - Using the application data of JBI software
  - **Across process domains**
    - flaw in protection domains

Attack Effects

- **Compromise**
  - Launching pad for further attacks
  - Malicious behavior
- **Crash**
  - Attack propagation stopped
- **Distinction between OSes with and without protection domains**

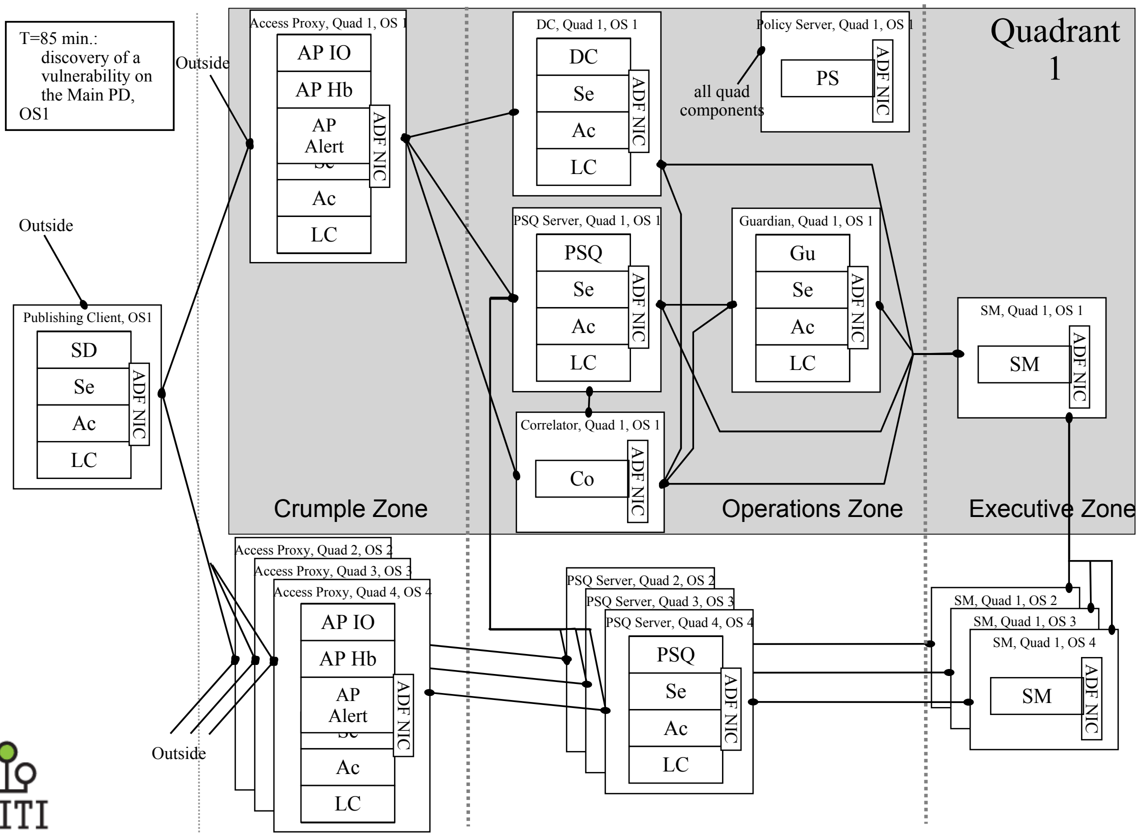
ITI

Attack Response

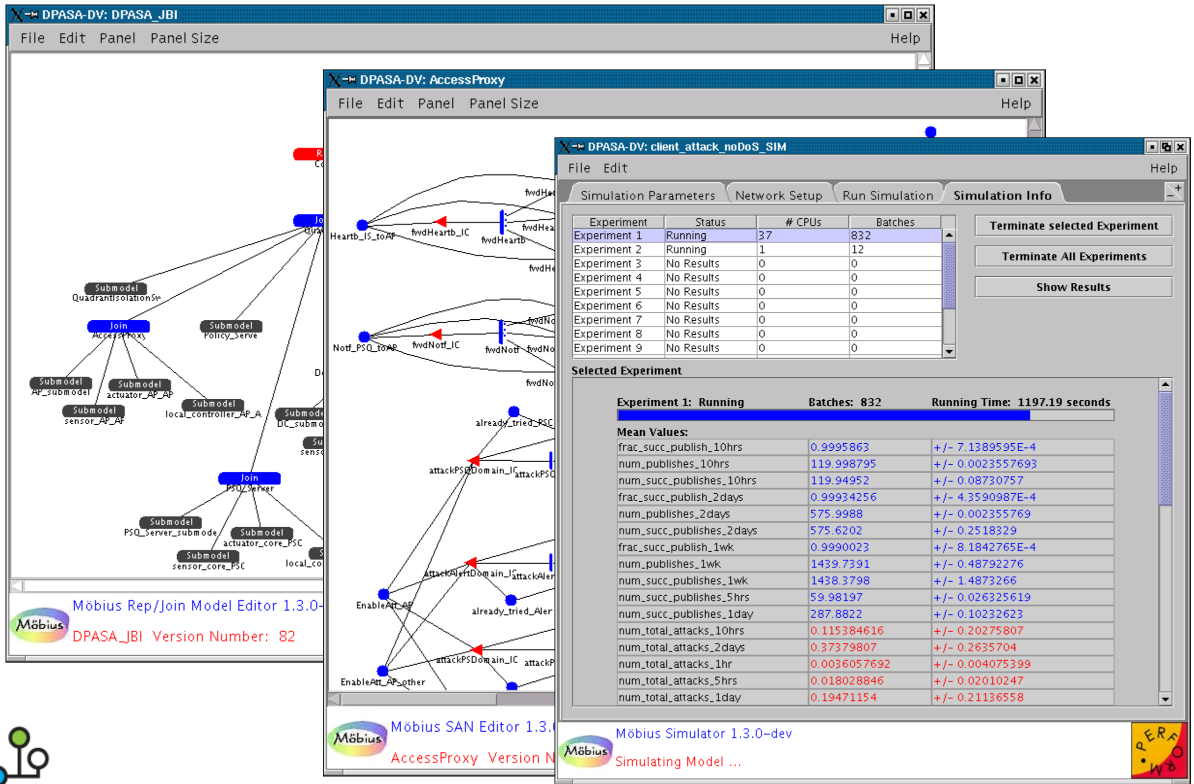
- **Intrusion Detection**
  - p_{detect}=0 if the sensors are compromised
  - p_{detect} > 0 otherwise.
- **Attack Responses**
  - Restart Processes
  - Secure Reboot
  - Permanent Isolation

ITI

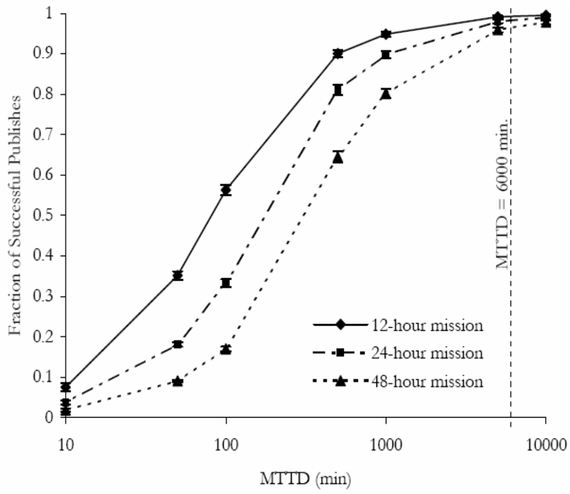
Infrastructure Attacks Example



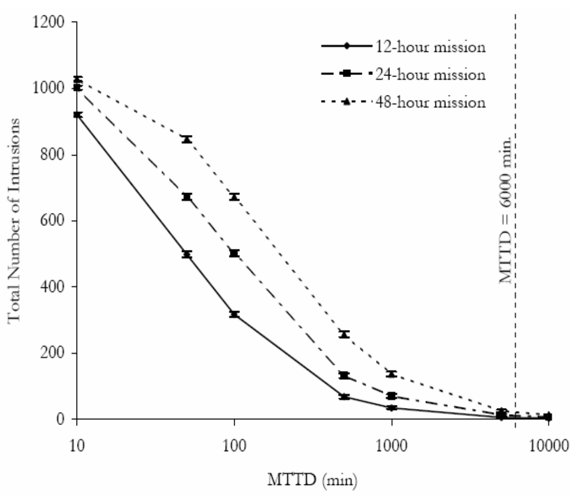
Construct Executable Functional Model



Vulnerability Discovery Rate Study

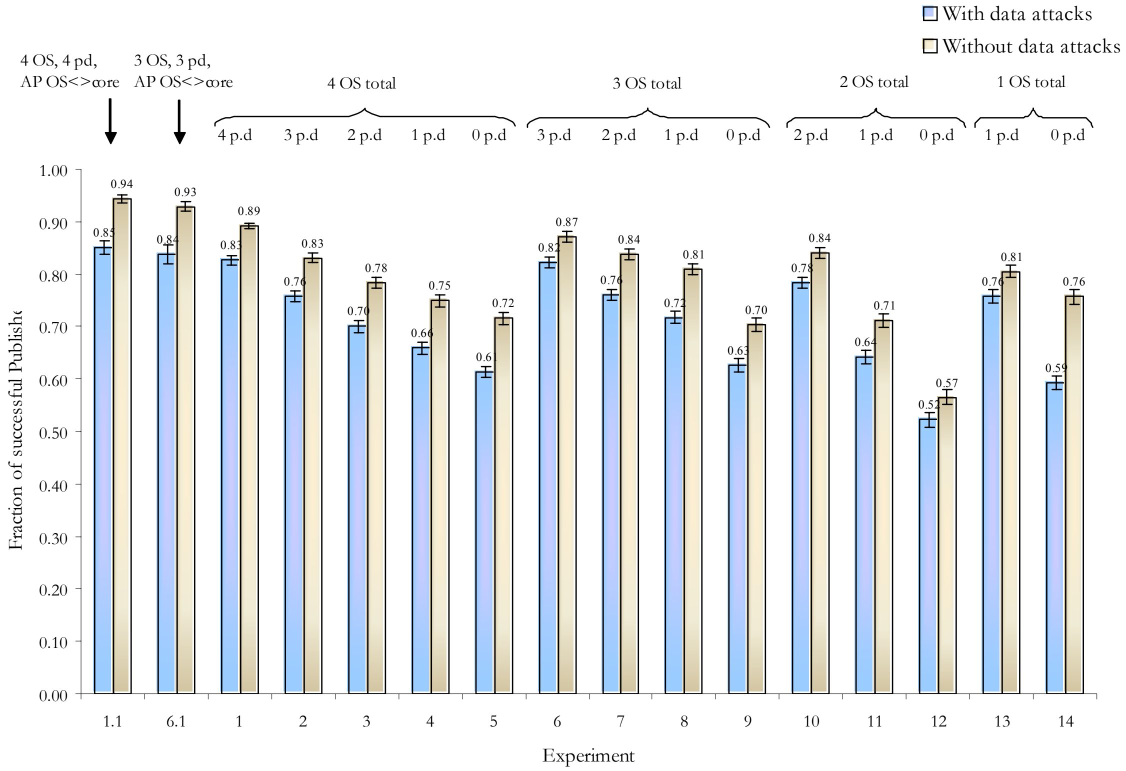


Fraction of successful publishes versus MTTD



Number of successful intrusions versus MTTD

Varying the number of OS and OS w/ process domains



Phase 2: Improving (and Validating) the Implementation

Objectives:

- Improve the system’s survivability
- Conduct specific system-level validation tasks
- Address all of the system-level concepts and mechanisms that may contribute to improvement, e.g., protocols and application scenarios

Main Idea:

- Think like an attacker
  - Examine whether a given attacker goal can be achieved
  - If so, alter the implementation so as to preclude such achievement

Procedure:

- Top-down, beginning with a specific high-level attacker goal
- Critical steps of the high-level attack tree are elaborated further as sub-trees, down to a level that admits adversarial testing.

ITI

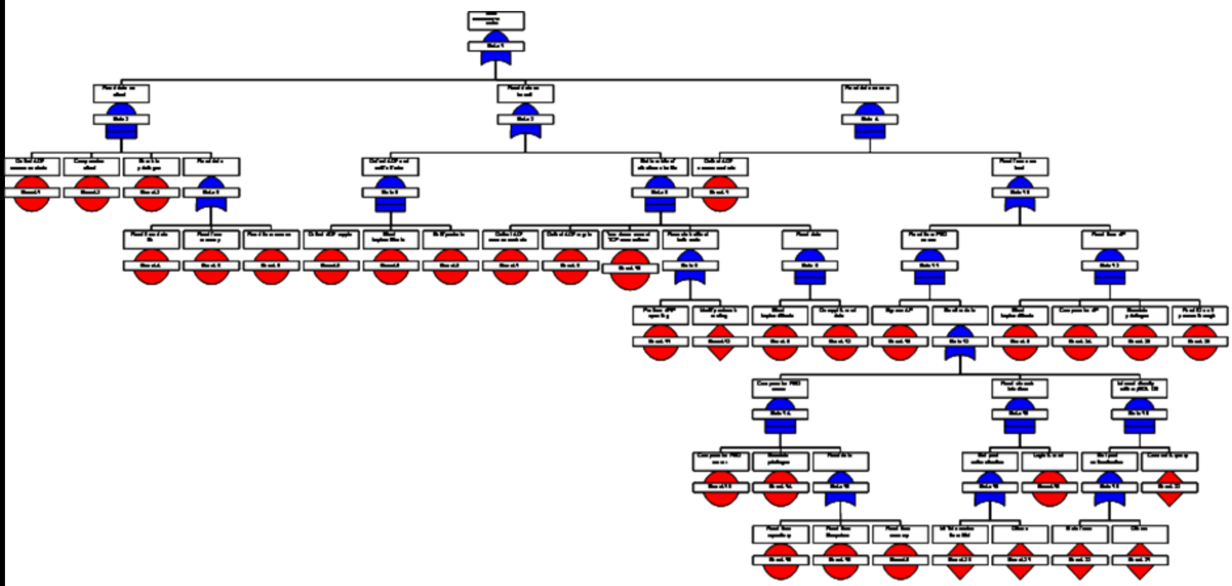
Attacker Goals

- We considered the following attacker goals:

G1: Prevent client publish
G2: Prevent IO delivery to client (Subscription)
G3: Prevent a successful query operation
G4: Prevent a successful client registration
G5: Defeat confidentiality of IO data
G6: Modify IO data
G7: Modify data in repository

ITI

G5: Defeat Confidentiality of IO Data



G5: Attack Steps/Minimal Attacks

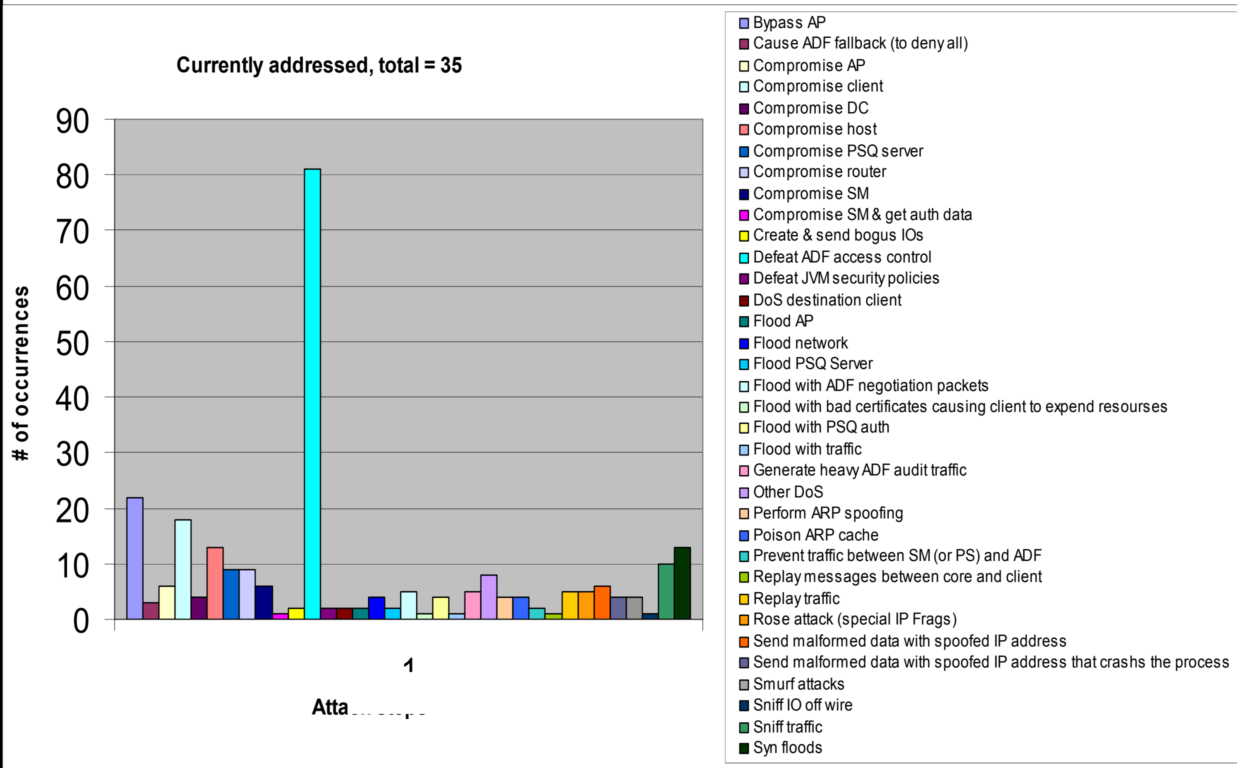
Attack Step #	Type	Attack Step Description		Minimal Attack Sets
1 (3)	BASIC	Defeat ADF access control		7 , 8 , 9
2	BASIC	Compromise client		5 , 3 , 2 , 1
3 (3)	UNDEVELOPED	Escalate privilege		4 , 3 , 2 , 1
4	BASIC	Read from data file		6 , 3 , 2 , 1
5 (2)	BASIC	Read from memory		16 , 21 , 19 , 1
6	BASIC	Read from screen		16 , 20 , 19 , 1
7 (2)	BASIC	Defeat ADF crypto		16 , 21 , 22 , 1
8 (3)	BASIC	Steal key/certificate		16 , 23 , 22 , 1
9 (2)	BASIC	Sniff packets		
10	UNDEVELOPED	Tear down current TCP connections		
11	BASIC	Perform ARP spoofing		
12	UNDEVELOPED	Modify network routing		
13	BASIC	Decrypt & read data		
15	BASIC	Compromised PSQ server		
16	BASIC	Bypass AP		
17	BASIC	Read from filesystem		
18	BASIC	Read from repository		
19	BASIC	Login & read		
20	UNDEVELOPED	MITM session from SM		
21 (2)	UNDEVELOPED	Others		
22	UNDEVELOPED	Connect & query		
23	UNDEVELOPED	Brute force		
24	BASIC	Compromise AP		
25	BASIC	Read IO as it passes through		

Summary of Attack Steps/Minimal Attacks

- For the seven high-level attack trees that were developed, there are
  - 524 attack steps (including repeats)
  - 114 different attack steps
- The number of different minimal attacks for each high-level goal (these are derived automatically from a goal's attack tree) are as follows.
  - G1: 54, G2: 43, G3: 36, G4: 52, G5: 8, G6: 12, G7: 11
- Total number of minimal attacks with respect to all goals: 216

ITI

Attack Steps Frequency of Occurrence



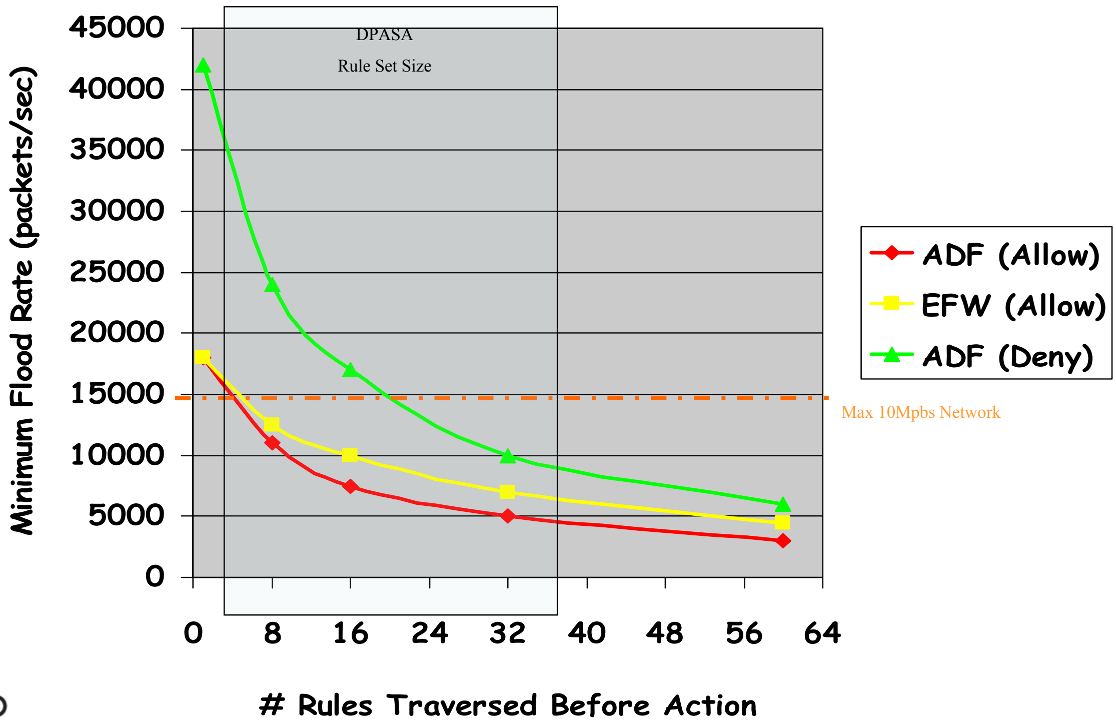
ITI

Example Attack Step Analysis: ADF DOS Attack

- Three Metrics were used to benchmark the ADF.
  - *Max. Throughput*: The fastest receive rate at which there is no packet loss
  - *Available Bandwidth*: The amount of data that can be transmitted in a fixed amount of time (when no flood in progress)
  - *Minimum Flood Rate*: The lowest rate of flood which leads to a successful denial of service attack.
- Floods cause packet loss, which in turn lowers bandwidth due to TCP congestion control. UDP will suffer high packet loss.
- Experimental Setup
  - Follows rfc2544 as much as possible
  - Max flood rate is ~44000 frames/sec = 22 Mbits/sec (for 64 Byte frames)

ITI

Minimum Flood Rate for Successful DoS on ADF NIC



ITI

Conclusions

- How can the behavior of attackers be quantified?
  - By their effect, if system is intrusion tolerant
- How should security/survivability measures be specified?
  - In terms of the definition of "proper operation" for the system
- If relative measures are desired, can they be shown to be robust across a wide variety of situations?
  - Yes, through extensive simulation
- How accurately can absolute measures be estimated?
  - Unknown ???
- Can quantification aid in security testing?
  - Yes, through (advanced) attack tree analysis
- Can a notion of "coverage" be developed for security testing?
  - Unknown ???

ITI