3rd Information Trust Institute Workshop
on Dependability and Security

*Panel 4 — Assessment*

University of Illinois at Urbana-Champaign, USA — Tuesday December 5, 2006

# Joining Efforts Towards Dependability & Security Assessment

**Jean Arlat**
(jean.arlat@laas.fr)

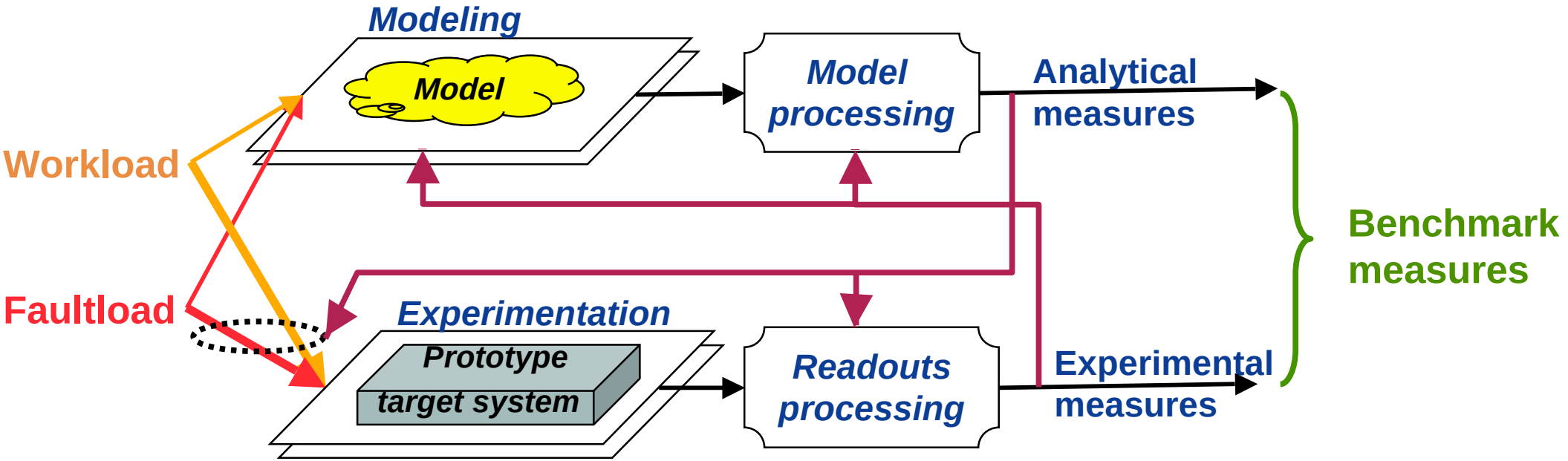
# Assessment

Modeling & Simulation, Controlled Experimentation, Field Measurement

- **Accidental faults: HW, SW, Operator**
  - Fault/Error models (stuck-at, bit-flip, ODC, etc.)
  - Probabilistic modeling (simulation, CTMC, SPN, SW reliability growth, etc.)
    —> Integration into the main design thread (UML, AADL)
  - Experimentation (field measurement, fault injection, dependability benchmarking, etc.)
- **Malicious faults: Insiders, Outsiders**
  - Threats and vulnerabilities
  - Evaluation criteria (TCSEC, ITSEC, CC) ≈ qualitative assessment
    —> quantitative assessment of operation security?
  - Experimentation (testing scripts, fault injection, honeypots, etc.)

# Dependability Benchmarking

**Agreement: Representativeness, Reproducibility, Portability, Cost Effectiveness, Scalability**




**2001-2004 — IST project DBench (LAAS-CNRS, Chalmers U., Critical SW, U. Coimbra, U. Erlangen, Microsoft, U. Valencia) [www.laas.fr/dbench]**

**1999->… — IFIP WG 10.4 SIGDEB (CMU, Critical SW, HP, IBM, Intel, LAAS-CNRS, Sun, U. Coimbra, UIUC, U. Valencia, etc.) [www.laas.fr/~kanoun/ifip_wg_10_4_sigdeb]**

**—> Book [K. Kanoun & L. Spainhower Ed., IEEE CS, 2007]**

# From Software Faults to Faultload (1/2)

- **IBM *Orthogonal Defect Classification***
  **A SW fault is characterized by the change in the code that is necessary to correct it**
  - **Fault trigger** — **Conditions that make the fault to become an error**
  - **Fault type** — **Type of mistake in the code**
    - **Assignment** — values assigned incorrectly or not assigned
    - **Checking** — missing or incorrect validation of data, or incorrect loop, or incorrect conditional statement
    - **Timing/serialization** — missing or incorrect serialization of shared resources
    - **Algorithm** — incorrect or missing implementation that can be fixed without the need of design change
    - **Function** — incorrect or missing implementation that requires a design change to be corrected
- **Typical Data Table**

| ID | open date | closed date | activity | trigger | impact | type | qualifier | source | age | severity |
|---|---|---|---|---|---|---|---|---|---|---|
| 12345 | 3/1/97 | 3/8/97 | des/rev | conformance | capability | assign | miss | in-house | new | 2 |
| 12377 | 6/1/97 | 6/15/97 | unit test | simple | usability | checking | miss | in-house | new | 2 |
| 12470 | 6/5/97 | 7/15/97 | function test | coverage | integrity security | algorithm | incorrect | outsourced | refixed | 1 |
| 12543 | 8/4/97 | 8/30/97 | system test | soft config | reliability | function | miss | ported | rewritten | 1 |

http://www.research.ibm.com/softeng/ODC/ODCEG.HTM#datatable

# From Software Faults to Faultload (2/2)

- **U. Coimbra: Study Failure Reports from 9 OSS Programs:**
  **(text editors, Linux kernel, game, etc.)**

| ODC Type | # of faults | ODC distribution (U. Coimbra) |
|---|---|---|
| Assignment | 118 | 22.1 % |
| Checking | 137 | 25.7 % |
| Interface | 43 | 8.0 % |
| Algorithm | 198 | 37.2 % |
| Function | 36 | 6.7 % |

- **Alternative Fault Classification —>**

  **Faults considered as language constructs that are:**
  - **Missing** (e.g., missing part of a logical expression)
  - **Wrong** (e.g., wrong value used in assignment)
  - **Extraneous** (e.g., extra condition in a test)

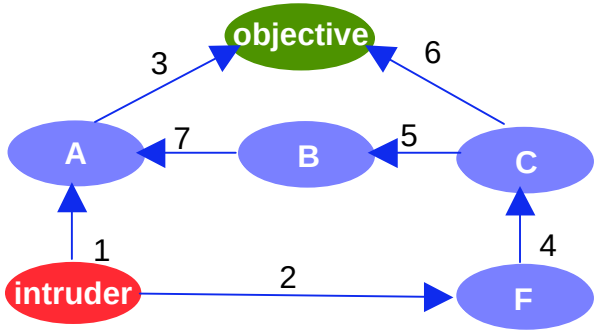
  **—> Propose a Mutation Strategy for machine-code level**

| ODC types | Nature | # faults |
|---|---|---|
| Assign. | Missing | 44 |
| | Wrong | 64 |
| | Extraneous | 10 |
| Check. | Missing | 90 |
| | Wrong | 47 |
| | Extraneous | 0 |
| Interf. | Missing | 11 |
| | Wrong | 32 |
| | Extraneous | 0 |
| Alg. | Missing | 155 |
| | Wrong | 37 |
| | Extraneous | 6 |
| Func. | Missing | 21 |
| | Wrong | 15 |
| | Extraneous | 0 |

Emulation of Software Faults: A Field Data Study and a Practical Approach,
J. Durães, H. Madeira, IEEE TSE, Vol. 32 No.11, Nov. 2006, pp. 849-867

# Quantitative Assessment of Security

## Vulnerabilities Modeling "privilege graph"

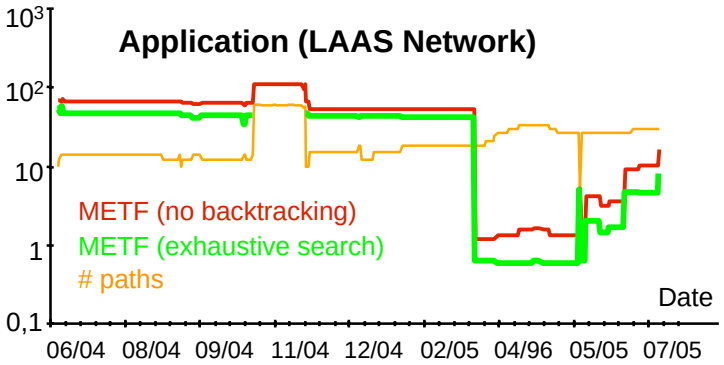



**Node =** set of privileges

**Arc =** vulnerability class

**Path =** sequence of vulnerabilities that could be exploited by an attacker to defeat a security objective

**Arc weight =** effort to exploit the vulnerability



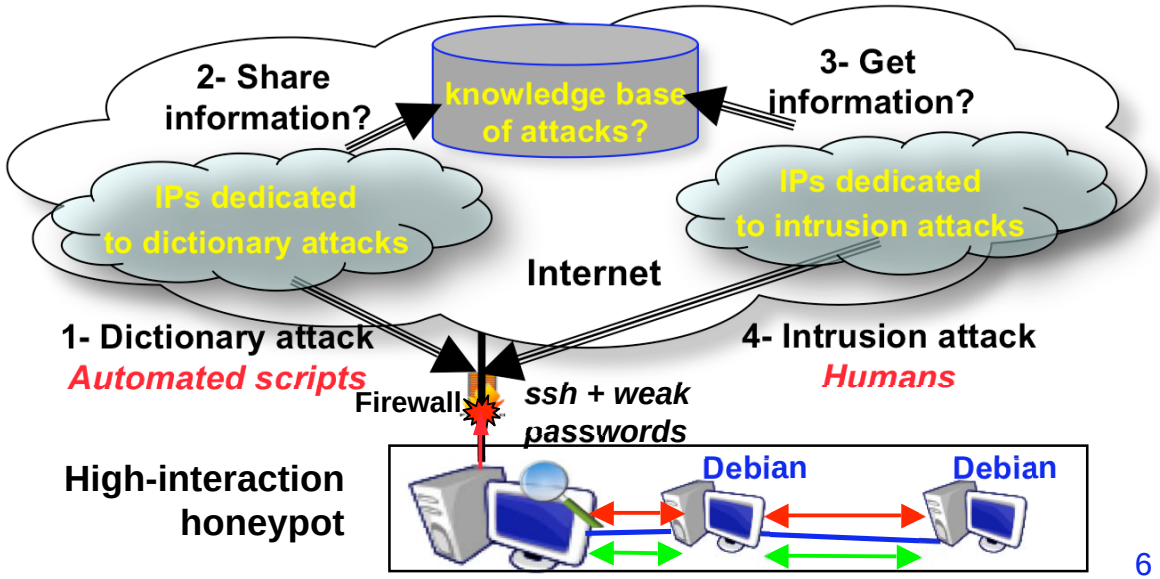

## -> Questions?

- Is such a model valid in the real world?
- Considered behaviors are two extreme ones, but, what would be a "real" attacker behavior?
- Weight parameters are assessed arbitrarily (subjective?)

## -> Wanted ! Real Data

**CADHo project: "Collection and analysis of Attack Data based on Honeypots (Eurecom, LAAS-CNRS, Renater)**

- Both low- (35 worldwide) and high-interaction honeypots
- Typical behavior:



# Some Concluding Remarks

- **Academia/Industry Cooperation: Some successful stories…**
  - -> LIS Laboratory for Dependability Engineering [www.laas.fr/LIS] 1992-2000
  - -> RIS Network for Dependability Engineering [www.ris.prd.fr] 2001-2004
    (LAAS-CNRS, Airbus, Astrium, EdF, Technicatome, Thales)
  - -> ReSIST Resilience for Survivability in IST (NoE) [www.resist-noe.org] 2006-2008
- **With few exceptions, industry reluctant to disclose fault/threat data (including contextual information)**
  - —> OSS community
  - —> Deployment of honeypots
- **Security assessment compatible with quantitative approaches?**
- **Some Additional Challenges Ahead:**
  - **System Features:**
    - Ubiquity, Evolvability, Openess, Scalability, Diversity
    - Network of Mobile Entities (**Hidenets**) [www.hidenets.aau.dk]
    - Interdependencies in Critical Infrastructures (**Crutial**) [crutial.cesiricerca.it]
  - **Assessment Techniques:**
    - Analytical Evaluation *and* Experimentation
    - Formal methods (Proving, Model Checking) *and* Testing
    - Accidental *and* Malicious Faullts