Intrusion Tolerance: Dependability vs. Security

David Powell
Yves Deswarte
LAAS-CNRS
Toulouse, France
deswarte@laas.fr

Who are the attackers/intruders?

1: Outsider

2: User

3: Privileged User

...because the least privilege principle is poorly implemented
“Classical” security

- Authentication
  - should prevent non-registered users to access the system

- Authorization
  - should prevent users to perform illegitimate actions

... but prevention is insufficient:

Mechanisms are imperfect:
- Authentication can be deceived or bypassed
  - Passwords can be guessed, cracked or disclosed
  - Tokens can be forged, cloned or stolen
  - Biometric sensors can be deceived...
    - ... and biometric information cannot be revoked!
- Authorization is difficult:
  - Security policy trade-offs: complexity/flexibility
  - Still more difficult for distributed systems
  - Protection mechanisms are inefficient and/or unreliable
- Flaws/malicious logic in implementation
Dependability

- Trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers


The Dependability Tree

- Dependability
  - Impairments
    - Fault
      - Error
      - Failure
  - Methods
    - Fault Prevention
    - Fault Tolerance
    - Fault Removal
    - Fault Forecasting
  - Attributes
    - Availability
    - Reliability
    - Safety
    - Confidentiality
    - Integrity
    - Maintainability

Security
The Dependability Tree

Dependability
- Attributes
  - Availability
  - Reliability
  - Safety
  - Confidentiality
  - Integrity
  - Maintainability
- Impairments
  - Fault
  - Error
  - Failure
- Methods
  - Fault Prevention
  - Fault Tolerance
  - Fault Removal
  - Fault Forecasting

Are these attributes sufficient?

Dependability
- Attributes
  - Availability
  - Reliability
  - Safety
  - Confidentiality
  - Integrity
  - Maintainability
- Impairments
  - Fault
  - Error
  - Failure
- Methods
  - Fault Prevention
  - Fault Tolerance
  - Fault Removal
  - Fault Forecasting

w.r.t. authorized actions

Security
Security Properties

- Availability
- Anonymity
- Privacy
- Integrity
- Authenticity
- Secrecy
- Confidentiality
- Accountability
- Non-repudiability
- Auditability
- Imputability
- Irrefutability
- Traceability
- Opposability
- Availability
- Anonymity
- Privacy
- Integrity
- Authenticity
- Secrecy
- Confidentiality
- Accountability
- Non-repudiability
- Auditability
- Imputability
- Irrefutability
- Traceability
- Opposability
Security Properties

- Confidentiality
- Integrity
- Availability

of

Information

Meta-information

Accountability - A+I
- existence of operation
- identity of person

Anonymity - C
- personal data

Privacy - C
- message content

Authenticity - I
- message origin

Non-repudiation - A+I
- sender, receiver identity

The Dependability Tree

Dependability
- Attributes
  - Availability
  - Reliability
  - Safety
  - Confidentiality
  - Integrity
  - Maintainability
- Impairments
  - Fault
  - Error
  - Failure
- Methods
  - Fault Prevention
  - Fault Tolerance
  - Fault Removal
  - Fault Forecasting
Fault, Error & Failure

**Error**
that part of system "state" which may lead to a failure

**Failure**
occurs when delivered service deviates from implementing the system function

---

Example: Single Event Latch up

SELs (reversible stuck-at faults) may occur because of radiation (e.g., cosmic ray, high energy ions)

Cosmic Ray
External fault

Lack of shielding

Vulnerability
Internal, active fault

SEL
Internal, externally-induced fault

Satellite on-board computer
Intrusions

Intrusions result from (at least partially) successful attacks:

- **Internal, dormant fault**
- **Intrusions**
- **Internal, active fault**
- **External fault**
- **Vulnerability**
- **Internal, externally-induced fault**

**Computing System**

**Fault Model**

- **Human malice**
- **Error**
- **Failure**
- **Social system**
- **Attack**
- **Vulnerability**
- **Authentication & authorisation system**
- **Relations**
  - "exploits"
  - "causes"
  - "is interpreted as"
Fault Types

- **attack** - malicious external activity aiming to intentionally violate one or more security properties; an *intrusion* attempt.
- **vulnerability** - an accidental fault, or a malicious or non-malicious intentional fault, in the requirements, the specification, the design or the configuration of the system, or in the way it is used, that could be exploited to create an *intrusion*.
- **intrusion** - a malicious interaction fault resulting from an attack that has been successful in exploiting a vulnerability.

Outsiders vs Insiders

- Outsider: not authorized to perform any of specified object-operations
- Insider: authorized to perform some of specified object-operations
The Dependability Tree

Dependability

Attributes
- Availability
- Reliability
- Safety
- Confidentiality
- Integrity
- Maintainability

Impairments
- Fault
- Error
- Failure

Methods
- Fault Prevention
- Fault Tolerance
- Fault Removal
- Fault Forecasting

Fault prevention = classical security

attacker

designer/operator

attack prevention

intrusion prevention

intrusion tolerance

vulnerability prevention

vulnerability

vulnerability removal

fault removal

failure
The Dependability Tree

- Dependability
  - Attributes
    - Availability
    - Reliability
    - Safety
    - Confidentiality
    - Integrity
    - Maintainability
  - Impairments
    - Fault
    - Error
    - Failure
  - Methods
    - Fault Prevention
    - Fault Tolerance
    - Fault Removal
    - Fault Forecasting

Fault Tolerance

- Fault
  - Error
  - Failure

- Error Processing
  - Detection
  - Damage assessment
  - Recovery

- Fault Treatment
  - Diagnosis
  - Isolation
  - Reconfiguration
Error Detection (1)

- Likelihood checking
  - by hardware:
    - inexistent or forbidden address, instruction, command...
    - watchdogs
    - error detection code (e.g., parity)
  - by software (OS or application) = verify properties on:
    - values (absolute, relative, intervals)
    - formats and types
    - events (instants, delays, sequences)
  - Signatures (error detection code)

Error Detection (2)

- Comparison between replicates
  - Assumption: a unique fault generates different errors on different replicates
    - internal hardware fault: identical copies
    - external physical fault: "similar" copies
    - design fault / human interaction fault: diversified copies

- On-line model checking
"Intrusion" Detection

Core concepts: AKA “built-in test”
- Memory scrubbing
- Software rejuvenation

Interpretation wrt. intrusions
- Vulnerability scanning
- Configuration checking
Damage assessment: containment regions

Error Recovery

Backward recovery

Forward recovery

Compensation-based recovery (fault masking)
Error Processing (wrt. intrusions)

- Error detection
  - + Backward recovery (availability, integrity)
  - + Forward recovery (availability, confidentiality)

- Intrusion masking
  - Fragmentation (confidentiality)
  - Redundancy (availability, integrity)
  - Scattering

Intrusion Masking

Intrusion into a part of the system should give access only to non-significant information

FRS: Fragmentation-Redundancy-Scattering

- **Fragmentation**: split the data into fragments so that isolated fragments contain no significant information: confidentiality

- **Redundancy**: add redundancy so that fragment modification or destruction would not impede legitimate access: integrity + availability

- **Scattering**: isolate individual fragments
Different kinds of scattering

- **Space:** use different transmission links and different storage sites
- **Time:** mix fragments (from the same source, from different sources, with jamming)
- **Frequency:** use different carrier frequencies (spread-spectrum)
- **Privilege:** require the co-operation of differently privileged entities to realize an operation (separation of duty, secret sharing)

Delta-4 Prototype

[Blain & Deswarte 1994]

[Deswarte et al. 1991]

[Fraga & Powell 1985]

[Fray et al. 1986]

[Fabre et al. 1994]
FRSed File Server

Fragmentation  Replication  Scattering

File  Fragments  

User Site  Multicast Network  Storage Sites

File Fragmentation

Fragment name := OWHF(file name, page #, frgt #, key)
FRSed Security Management

- No single trusted site or administrator
- Global trust in a majority of security sites (and administrators)

Authentication

1. Smartcard Activation
2. Local Authentication
3. Global Decision
4. Session key distribution
Authorization

1. Request to open a session
2. Global Decision
3. tickets
4. direct access

Security Server

File name | ACL | Key shadow
---------|-----|-------------

Secured Servers

Fragmented Data Processing

Fragmentation

Replication

Scattering

Site 1

Site 2

Site 3

Site 4
Fault Tolerance

Error

Failure

Fault Treatment

Diagnosis
- determine cause of error, i.e., the fault(s)
  - localization
  - nature

Isolation
- prevent new activation

Reconfiguration
- so that fault-free components can provide an adequate, although degraded, service

Error Processing
- Detection
- Damage assessment
- Recovery
Fault Diagnosis

- **Intrusion diagnosis**, i.e., trying to assess the degree of success of the intruder in terms of system corruption
- **Vulnerability diagnosis**, i.e., trying to understand the channels through which the intrusion took place so that corrective maintenance can be carried out (diagnosis immediate if errors signaled by vulnerability scanner or configuration checker)
- **Attack diagnosis**, i.e., finding out who or what organization is responsible for the attack in order that appropriate litigation or retaliation may be initiated

Fault Isolation

- **Interpretation wrt. intrusions**
  - Blocking traffic from an intrusion containment domain that is diagnosed as corrupt, by, for example, changing the settings of firewalls or routers
  - Removing a corrupted file from the system
- **Interpretation wrt. root causes** (vulnerability/attack)
  - Taking off line software versions with newly-found vulnerabilities
  - Arresting the attacker
System Reconfiguration

- Interpretation wrt. intrusions
  - Change a voting threshold, e.g., 3/5 => 2/3 after 2 corruptions
  - Deployment of countermeasures, inc. probes and traps
- Corrective maintenance actions
  - Vulnerability removal
    - software revision and upgrade
    - security patches
  - Attacker rehabilitation

The Dependability Tree

- Attributes
  - Availability
  - Reliability
  - Safety
  - Confidentiality
  - Integrity
  - Maintainability
- Impairments
  - Fault
  - Error
  - Failure
- Methods
  - Fault Prevention
  - Fault Tolerance
  - Fault Removal
  - Fault Forecasting
Fault removal

The Dependability Tree

- **Attributes**
  - Availability
  - Reliability
  - Safety
  - Confidentiality
  - Integrity
  - Maintainability

- **Impairments**
  - Fault
  - Error
  - Failure

- **Methods**
  - Fault Prevention
  - Fault Tolerance
  - Fault Removal
  - Fault Forecasting

Security
Fault forecasting

- **Evaluation**:  
  - Gain confidence that system dependability is satisfactory  
  - Select architecture/components to achieve the best dependability-performance-cost trade-off

- **Quantitative measures**:  
  - **Reliability**: $MTFF = \text{mean time to first failure}$,  
    $R(t) = \text{prob}_{\text{continuous service}}(t)$  
  - **Availability**: $MTBF/(MTBF+MTTR)$,  
    $A(t) = \text{prob}_{\text{correct service provided when needed}}(t)$

---

Basic assumption

- **Faults** = elementary component failures (or other rare physical phenomena)  
- **Model** = independent stochastic processes with ~uniform distribution

- **OK** for physical H/W faults and most environmental faults

- **~OK** for most S/W design faults (bugs)

- **Not OK** for attacks or malicious design faults
Security Evaluation

- Usual techniques
  - Evaluation criteria (TCSEC, ITSEC, CC, ...): ~ qualitative evaluation
  - Risk assessment: subjective evaluation of vulnerabilities, threats, consequences
  - These are static analyses rather than dynamic: "How the system has been built?" rather than "How is it operated?"

Quantitative security evaluation

- Measure = effort needed for a possible attacker to defeat the security policy

- Objectives:
  - Take into account security/usability trade-offs
  - Monitor security evolutions according to configuration and use changes
  - Identify the best security improvement for the least usability change
General approach

- Identify security objectives: security policy
- Model (operational) system vulnerabilities
- Model the attack processes
- Compute significant measures

Vulnerability modeling

Privilege graph

- Node = a set of privileges (user, group, rôle, ...)
- Arc = a method to transfer privileges = vulnerability
- Path = set of vulnerabilities usable by a possible attacker to reach a target
- Weight = for each arc, effort to exploit the arc’s vulnerability

1) X can guess Y's password
2) X can install a Trojan horse that Y can activate
3) X can exploit a flaw in Y's mailer
4) Y is a subset of X
5) Y uses a program that X can modify
6) X can modify a "s-uid" program owned by Y
7) X is in Y's .rhosts
Assumptions on the attack process

- Attack process = all possible successful attack scenarios
- Reasonable assumptions
  - The attacker knows only the vulnerabilities that can be exploited with the privileges he already owns.
  - The attacker will not exploit vulnerabilities which would give him privileges he already owns.
- Plus one out of the two following assumptions:
  - Total Memory (TM): the attacker remembers all the vulnerabilities he did not exploited in the previous steps, and he can "back-track".
  - Memory-Less (ML): the attacker considers only the vulnerabilities that can be exploited with the new privileges he just acquired.

Attack Process Examples

ML Assumption

TM Assumption
Measure computation

1. Identify the attacker-target couples

2. For each couple, compute:

   **METF-ML**: Mean Effort To security Failure (i.e. to reach the target) with ML assumption.

   **METF-TM**: Mean Effort To security Failure with TM assumption.

   **Shortest Path**: Mean effort to go through the shortest path.

   **Number of Paths**: Number of possible paths from the attacker to the target nodes.

---

ESOPE Tool Set

(Évaluation de la Sécurité OPERationnelle)
Experiment report

Objectives:
- **Validate the approach:**
  - Assess the measure pertinence **wrt.** system changes (configuration, users, ...)
  - Feasibility of a full-size system evaluation.
- **Was not aimed:**
  - Correct the identified vulnerabilities

Experiment context

**Target system:**
- Unix
- 700 users - 300 machines - LAN
- 13 months (June 1995 - July 1996)

13 types of vulnerabilities (files `.rhosts`, `.*rc`, passwords, etc.)

**Security objectives:**

<table>
<thead>
<tr>
<th>Objective</th>
<th>Attacker</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective 1</td>
<td>insider</td>
<td>root</td>
</tr>
<tr>
<td>Objective 2</td>
<td>insider</td>
<td>admin_group</td>
</tr>
</tbody>
</table>

4 difficulty levels:

<table>
<thead>
<tr>
<th>Type</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>immediate</td>
<td>10</td>
</tr>
<tr>
<td>easy</td>
<td>(10^2)</td>
</tr>
<tr>
<td>difficult</td>
<td>(10^3)</td>
</tr>
<tr>
<td>very difficult</td>
<td>(10^4)</td>
</tr>
</tbody>
</table>
Comparison between measures

- The shortest path (SP) is not sensitive enough to identify important events.
- The number of paths (NP) changes too often and would produce a large number of false alarms.
- METF-ML presents a good sensitivity to important events.
- METF-TM is easier to interpret, but is sometimes too complex to be computed.

References