CSED601
Dependable Computing
Lecture 14

Jong Kim

Dept. of CSE

POSTECH

References

- http://www.laas.fr/IFIPWG/Workshops/44/W1/06-Littlewood.pdf (ESORICS 2004 : Lecture Notes in Computer Science Volume 3193, 2004, pp 423-438)
- http://www.laas.fr/IFIPWG/Workshops/42/06-Schlichting.pdf

Redundancy

- Traditional fault tolerance
  - Time redundancy: repeated execution, retransmission
  - Space redundancy: replication of data/computation
- Both can be (and have been) used to increase survivability
- Redundant methods: Use two or more methods to enforce a security property
- Goal: Properties ensured through redundancy should remain valid even if some of the methods used have been compromised.

Example

- Confidentiality in communication security
  - Successive encryption with different methods
  - Alternating order of methods used
  - Apply different methods to different message in a stream
- Authentication
  - Two or more independent authentication services (e.g., PKI, Kerberos)
  - Multiple user authentication methods (password, biometrics)

Impact of redundancy

- Eliminate single point of vulnerability
- Introduces artificial diversity into the system
- Introduces unpredictability

Role of Independence

- Redundancy increases survivability only if methods are independent, i.e., breaking one does not make it easier to break others
- Analogous to failure independence in fault tolerance
- Example: Sources of dependency in communication security
  - Same key used by different methods
  - Same key creation/distribution method used
  - Keys are stored in the same place
  - Methods of combining encryption algorithms
  - Etc.

Independence Enhancement

- Techniques to increase independence
  - Use different keys established using different key distribution methods (e.g., Diffie-Hellman and Kerberos)
  - Unrelated encryption methods, e.g., different block sizes
  - Combination techniques that increases independence
- Redundancy can also be used for integrity, i.e., multiple message signatures

Redundant methods in other services

- Redundancy for PKI and certificate agencies
- Redundancy in file access control
  - Encrypted files (user must be both authorized and have the key),
  - Monitoring for changes to important files (e.g., web pages, log files)
- IDS viewed as a redundant “failure detection” service

Redundancy & Diversity

- Redundancy: simple replication of a component in identical copies, as adopted against random hardware failures
- Diversity: different copies for the same purpose, used especially for multiple version software, in which redundant software versions are deliberately made to be different

Diversity Types

- Separation: designers simply seek to isolate redundant subsystems from as many as possible common potential causes of failures
- Forced diversity: trying to diversify the way the unavoidable common influences affect the redundant subsystems using different, functionally equivalent components in redundant subsystems
- Tailored diversity: if the designers know in some detail how their precautions affect susceptibility to causes of failures, this allows them to focus ‘forced diversity” for lower correlation between the effects of common influences on the redundant subsystems

What do diversity models do?

- Informally, diversity is clearly ‘a good thing’
  - But ‘independence’ is not believable, so simple mathematical calculations of efficacy are not available
  - Therefore issue is ‘how good is it’
    - How reliable will a diverse system be?
- Why can’t we claim independence? What is the nature of dependence?
  - Models for software diversity by Eckhardt and Lee, et al., gave insight into this via difficulty function

Difficulty function

- Idea here is that inputs to a program vary in ‘difficulty’
  - Here ‘difficulty’ can be thought of as ‘propensity to failure’
  - More precisely, $\theta(x)$, the chance that a program fails on x, is a function of x (the input being executed)
  - Some inputs are intrinsically harder to execute correctly than others
- Eckhardt and Lee model says ‘independent programs fail dependently’
  - Program A fails on a randomly selected input – this means it’s probably a ‘hard input’ – therefore increased chance that B will also fail

Difficulty function (2)

- Littlewood and Miller model generalizes this to 'forced diversity'
  - Design method A tries to overcome (some of) weaknesses of method B
  - In the ideal case, inputs that are difficult for program A will be easy for B and vice versa
- Detailed mathematics depends upon first and second moments of the random variables $\theta_A(X)$ and $\theta_B(X)$
  - Probability of failure of a 1-out-of-2 system is $E(\theta_A(X))E(\theta_B(X)) + Cov[\theta_A(X),\theta_B(X)]$
  - First term here is naïve 'independence' result

What can security learn form these?

- Dependence of failures between versions comes from subtle interplay between A and B difficulty variation
  - Apply this to diverse intrusion sensors?
    - Difficulty function over intrusions?
  - Qualitative results presumably carry over
    - E.g., ‘independent’ intrusion sensors do not show independent failures
    - Is anyone in security community assuming they do..?
  - Can we get a handle on quantitative efficacy of single detectors, and of dependence between them?

Example

- Suppose you have n possible intrusion sensors
- You want to use the most effective m-fold diverse detector (m < n)
- How do you select the m sensors to use?
- This decision requires knowledge of the individual efficacies of the sensors and of dependencies between them
- Diversity models would help here if we know the model parameters

Some novel modeling issues

- Do models apply also to diversity of intruders?
  - How do you pick the best red team of size m from n?
  - Ditto selection of intrusion procedures
- Diverse intruders with diverse sensors
  - Can we model this? Not generally an issue in reliability: nature does not mount diverse threats… does she?
  - How do the two types of diversity interact?
- Issue of false alarms – can models be extended to deal with trade-off here (sensitivity versus specificity)?
  - This has not been done in reliability versions of the models, surprisingly (it's needed there too)

Availability of data is crucial

- Real data needed
  - Historical? Honey-traps?
- How ‘strong’, how ‘diverse’, are real systems?
- Can we estimate from data the parameters of our models?
- Validation issues
  - Can we check our models against reality?
  - Can we check model predictions against reality?
  - Can we learn, and improve, from feedback?

Adaptation

- Adaptation: Changing execution behavior dynamically
- Two types
  - Value adaptations and algorithm adaptations
  - Changing parameters vs. changing methods
  - Both useful for survivability
- Predictive: Adapt methods when attack anticipated
- Reactive: Adapt compromised methods if an attack detected (e.g., IDS)
- Preventive: Adapt methods and parameters non-deterministically at runtime to increase artificial diversity and unpredictability

Impact and Caveat

- Impact:
  - Introduces artificial diversity into the system
  - Introduces unpredictability
  - Provides an approach for dealing with detected intrusion attempts
  - Provides an approach for graceful degradation
  - Provides an approach for dealing with changes in user security requirements
- Caveat:
  - Adaptation mechanisms must not make the service more vulnerable by introducing new attack modes

System support

- Issue: What kind of system support needed to build survivable services based on redundancy and adaptation?

Conclusions

- Redundancy and adaptation techniques can be used to increase the survivability of services
- Independence is a key requirement