On Bayesian Trust and Risk Forecasting for Compound Systems

7th International Conference on

IT Security Incident Management & IT Forensics

Nuremberg | 12th - 14th March 2013

Stefan Rass, Sebastian Kurowski

Contents

- Introduction – Motivation
- Basic Trust Model
  - Beta-Reputations
  - Updating Trust
  - Handling Uncertainty via Model Averaging
- Predicting Trust and Risk
- Prototype Implementation
- Conclusion and Future Steps

Introduction – Motivation

- Despite much efforts: trust still not well formalized
- One possibility that admits good interpretation of trust value: beta-reputation.
- Idea: interpret trust as the

  likelihood of correct behavior

- …based on frequentistic definition of probability:

$$\text{trust} = \frac{\text{number of cases in which the system functioned correctly}}{\text{number of all cases}}$$

- Where to get this information from:
  - Notifications of security incidents (tickets, incident documentation)
  - RSS feeds, scientific media
  - Experience
  - …

Motivation

- Security incidents are often well-documented (at least inside the enterprise perimeter)
- Trust in the system is established based on such experience
- Trust is an ingredient to decision making and risk management processes, but only „qualitatively“
- Why not use the information to reach a quantitative trust measure that can be used as a benchmark?
- Motivation of this research:

Combine incident documentation systems with a mathematical model of trust to use documentation for a well-established quantitative trust estimate of the overall system

Beta Reputations

Based on the understanding of trust as

$$\text{trust} = \frac{\text{number of cases in which the system functioned correctly}}{\text{number of all cases}}$$

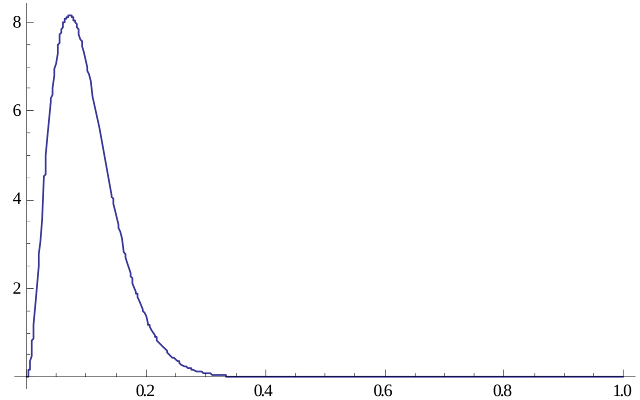
How to start? How do we incorporate new information?

Answer: Bayesian updating

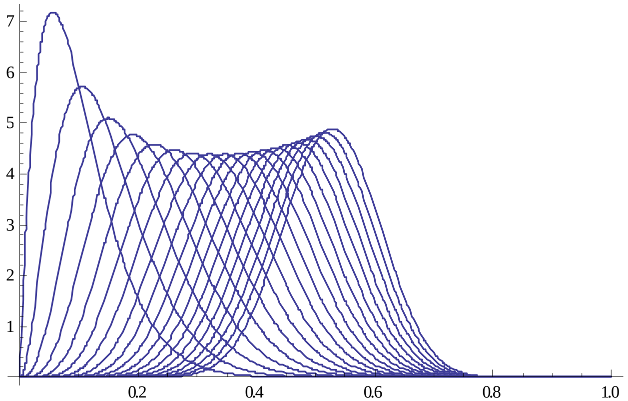
- Start: Beta-distribution (prior; based on expertise and experience)
- Updating: apply Bayes‘ rule to the so-far existing trust model to incorporate incoming information
- Advantage of using Beta-distributions:
  - Trust model remains a beta-distribution (mix) after the update
  - Updating is computationally efficient (and easy)

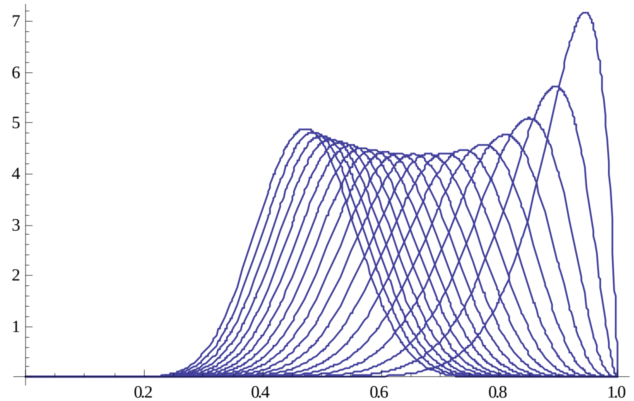
Beta distribution – Examples

- Original trust, say 90%. Prior specified as beta-distribution with parameters a, b such that $0.9 = \frac{b}{a+b}$



- Changes after a sequence of updates to account for
  a (negative) or b (positive) incidents:





expectation $\to$ 0 (full distrust) expectation $\to$ 1 (full trust)

Updating trust 1

- Simple approach:
  - Classify a notification as positive/negative and relevant/irrelevant for a particular component
  - Apply a Bayes-update for the component by either increasing a to $a+1$ (negative update) or b to $b+1$ (positive update)
- Take trust as the expectation of the beta-distribution, which is

$$
\begin{aligned}
E(Beta(a,b)) &= \frac{b}{b+a} \\
&= \frac{\text{\# of cases in which the system functioned correctly}}{\text{\# of all cases}} \\
&= \text{trust}
\end{aligned}
$$

- Problems: Classification is vague/uncertain
  $\rightarrow$ Bayes' rule does not directly apply

Updating trust 2

- Solution: Take the likelihoods as provided by the classifier:
  - Likelihood p_1: Does the notification really apply to this component?
  - Likelihood p_2: Is the update really negative? (otherwise positive)
  - The two can be regarded as independent, hence the Bayes‘ update itself applies with probability $p = p_1 p_2$
  - Do model averaging:

    new model = $p \times$ (updated model) + $(1-p) \times$ (current model)

- Bayes‘ updating and model averaging yields to a mix of beta-distributions, of size $O(n^2)$ after n updates (regardless of positive or negative)
- Trust value is the expectation of the resulting mix-distribution
- Interpretation as „trust = likelihood of misbehavior“ remains intact.

System Trust Model 1

- Next problem: this applies only to a single component!
- How to capture the system component‘s interplay?
- Solution:
  - Assign an indicator variable X_i to each component i, with
    $$X_i = \begin{cases} 1 & \text{if the component functions correctly,} \\ 0 & \text{otherwise} \end{cases}$$
  - Assign the same indicator X_{system} variable to the system (determining its value through the unknown interaction of components)
  - Each X_i determined by a mix of Beta-distributions f_i (component trust models)
  - Joint distribution of X_{system} determined by a copula-function C,
    $$X_{system} \sim C(f_1, \dots, f_n)$$
  - Trust in the system: again the expectation of X_{system}.

System Trust Model 2

- Where to get the copula from?
- Good news: we actually do not need it!
- Every copula satisfies the upper Frechét-Hoeffding-Bound:

$$C(x_1, \dots, x_n) \leq \min\{x_1, \dots, x_n\}$$

…elsewhere, e.g., in IT security management, known as maximum principle: the system is only as good as its weakest component!

- Hence, we are even statistically (mathematically) permitted to take the trust in the overall system as the minimal trust among all its components.

Predicting Trust and Risk 1

- Trust prediction: …by simulating positive and negative experience
- For a worst-case scenario analysis:
  - Simulate only negative incidents
  - …all 100% relevant to the system, resp. its components
- Multivariate non-smooth non-convex optimization problem (solveable by Nelder-Mead algorithm):

  minimize the total number of negative updates to components $1, 2, \ldots, n$*, subject to the system trust (minimum of all component trust value) is ≤* ε *(trust threshold).*

- Result: a sequence of negative trust updates to a set of components that would decrease the trust below a chosen threshold.

Predicting Trust and Risk 2

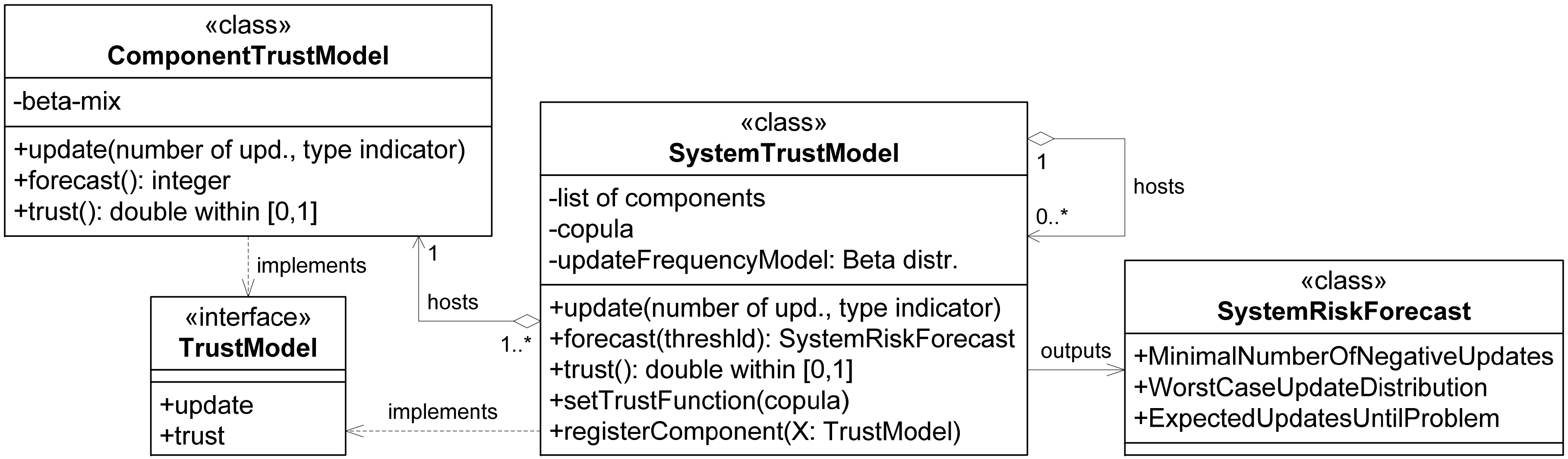
- When is the earliest point in time when such a scenario can possibly happen?
- Answer: use the predictive distribution
  - Minimal total number N of negative experience known from optimization problem ($\to$ pessimistic view)
  - Negative binomial (NB) distribution: counts the number of trials until N "successes" (negative incidents)
  - Predictive distribution: expected value of the NB distribution, based on our so-far recorded temporal update frequencies.
- Closed expressions can be given thanks to the beta-distributions: if $a^* > 1$ and $b^* \geq 0$ count the total number of updates, then it takes expectedly

$$N \cdot \frac{b^*}{a^* - 1}$$

updates before the worst-case scenario can become reality.

Prototype implementation

- Implemented the whole process (beta distributions, updating, model averaging, solving the nonlinear optimization problem) in a Java prototype
- Integration into Konstanz information miner (KNIME) currently under development
- Simplified prototype architecture diagram:



Experimental evaluation 1

- Time to compute a forecast depends on the size of the system (number of components) and the number of so-far recorded updates (determines the size of the beta-mix trust models)
- Empirical findings about the forecasting time under different settings:

Components	Updates	Forecasting time
fixed to 9	vary; $n = 10 \ldots 110$	$O(n^{1.67})$
vary; $n = 3 \ldots 30$	fixed to 50	$O(n^{2.64})$

- Exact asymptotic complexities are unknown, since the Nelder-Mead algorithm takes random starting values (possibly getting stuck at local optima); alternatives to be tested…

Experimental evaluation 2

- Sensitivity: how strongly is the trust affected by incoming information?
- Three kinds of questions:
  - How many negative updates would completely destroy trust?
  - How many positive updates are needed to recover from this?
  - How many positive updates are needed to gain almost full confidence?
- Experimental finding (data in the paper):

If an initial trust is destroyed upon negative experience, then it takes about an equal lot of positive experience to outweigh the doubts. However, it takes about 10 times as much positive experience to gain full confidence out of full distrust…

Conclusions

- Trust model has various degrees of freedom (design rationale included easy interpretation but also technical simplicity)
- Empirical evaluation needs to be more extensive and under real life conditions (so-far, the model performed very well under lab conditions)
- Integration into a standard decision support system desirable (and currently in progress)
- Prototype admits hierarchical modeling of systems. However, trust estimate is pessimistic; likelihoods will probably overestimate the true risk situation
- More precise estimates are achievable by a more accurate interplay model than the maximum principle (prototype supports user-defined copulas, however, it is unclear how to accurately model them)

Thanks for your attention

Questions?

Contact
stefan.rass@aau.at
sebastian.kurowski@iao.fraunhofer.de