The Trust Viewpoint

Rich Hilliard
<richh@mit.edu>

VERSION $2a$

Abstract

This is a work in progress. The author would be very interested to receive comments!

1 Introduction

This document defines a Trust Viewpoint for architecting trusted systems; it may also be known as: Security Viewpoint.

This architecture viewpoint is documented in accordance with ISO/IEC/-IEEE 42010, *Systems and software engineering — Architecture description* [6]. In particular, the requirements on viewpoints are found in Clause 7 of that Standard.

Keywords: trust, security, architecture, viewpoint

Version History

rev 2a 19 September 2014, Moved bibliography from `bibtex` to `biblatex`. Released revision with minor formatting fixes.

rev 2 13 May 2012, Moved document from Word to LaTeX. Removed 'rubrics' on how to document a viewpoint, these are now available separately as: Architecture Viewpoint template. Added boilerplate for stand-alone document. Made minor updates to text and organization to match published version of Standard and to follow the template. Based on comments received against rev 1, added new material on VP and MK operations.

rev 1 12 March 2009, initial release.

License

Comments or Questions

Contact the author (Rich Hilliard) with questions or comments.

2 Overview

> Trust is "Firm belief in the reliability, honesty, veracity, justice, strength, etc., of person or thing; condent expectation (that)." [*The Concise Oxford English Dictionary*, 1976]

Trust concerns pervade many systems[1] in a variety of ways, including security, secrecy, privacy, safety, and assurance.

The Trust viewpoint is intended to assist architects in the design, analysis and expression of trusted system architectures. It frames a number of architecture concerns related to trust and defines models for identifying threats and capturing decisions pertaining to the trustworthiness of the system of interest and the system response to identified threats. These decisions delineate security policies and the measures and mechanisms for attaining the needed levels of trust to counter anticipated threats. It is important to address trust as a part of architecture to insure that trust properties of the system can be achieved and so that solutions do not compromise other desired system properties and qualities.

[1] As used in this document, *system* is intended as a placeholder to refer any enterprise, software product or service, system of systems, or other "system of interest".

There is a large literature on trust and security in systems; it is not the objective here to replace or even adequately reference that literature. Rather, the objective here is to provide a small conceptual framework for trust such that Architects may utilize that body of existing work in a coherent fashion for their practice.

It is anticipated that Architects select, develop or adopt additional model kinds (or perhaps additional viewpoints) to apply in concert with the Trust Viewpoint when addressing specialties (such as those listed above).

A Trust view developed by the Architect is just one part of the trust and security work necessary to a system, which extends throughout the life cycle. The Trust Viewpoint here is to support that within the Architecting, resulting in inputs to subsequent security and design work; it is not intended as the total approach to trust for any system of interest.

3 Stakeholders and concerns

This section identifies Stakeholders: *Who are the audiences for Trust views?* and Concerns: *What will stakeholders find addressed by Trust views?*

Architects looking for an architecture viewpoint suitable for their purposes often use the identified concerns and the typical stakeholders to guide them in their search.

3.1 Trust concerns

Threats: What threats must the system counter? What risks are involved? How severe are those risks?

Confidentiality: Who gets to see what?

Integrity: Who can perform what actions? How are information and resources used, changed or updated?

Availability: Can information and resources be accessed when needed? How is timely access to information and resources achieved?

Measures and mechanisms: How are threats avoided, detected and mitigated? What are the solutions to meeting threats? How are they deployed? How do they interact with other architecture elements?

Policies: What security policies are to be enforced? How are they implemented?

Accountability: How are trust-relevant events monitored and recorded? What trusted (and other) system elements and action are audited?

3.2 Trust stakeholders

A Trust view resulting from applying this viewpoint may be of interest to the following stakeholders:

- Users
- Operators
- Owners
- Acquirers
- Accreditors
- Analysts
- Developers
- Suppliers

4 Trust model kinds

The Trust Viewpoint employs three model kinds (MK): the Threat MK, the (optional) Risk Assessment MK and the Security MK. The Threat MK is used to capture the type and nature of threats a system may face. The Risk Assessment MK is used to characterize the risk and severity of threats. The Security MK is used to express the major "trusted" elements of the architecture needed to address the threats, and given the risks.

5 Threat MK

A *threat model* identifies *threats* (or *hazards*[2]) against the system and the targets of those threats, called *resources* or *assets*. See Security MK. A resource has one or more *vulnerabilities*. Threats may exploit those vulnerabilities.

For each identified threat, the threat model should capture a name and description of the threat; its originating source or cause (when knowable); the target and its vulnerabilities; and the intent or objective of such an attack. Threats may be prioritized, if there are a large number, using expected Risk or various other heuristics.

A threat model can be captured in a table. A threat matrix template for this model is provided below.

5.1 Threat MK template

5.2 Threat MK operations

Identify Threats. Based on security objectives, known requirements (including availability and QoS requirements), compliance obligations (applicable security

[2]**hazard:** "A system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to an accident (loss)." [9]

Table 1: Threat matrix template

Threat Id	*A name or id for the threat*	
Description	*A brief description; with background or history if it is a known threat*	
Source, Cause	*From what does the threat originate?*	
Target, Vulnerabilities	*What resources does this threat target?* *What vulnerabilities of that resource does the threat attempt to exploit?*	*This may not be determinable without reference to other elements of the architecture.*
Intent	*What is the intent of the threat? (e.g., denial of service, destruction of data, fraud)*	
Risk	*Refer to Risk assessment MK.*	*Use for threat prioritization, if needed.*

policies, laws, standards), identify the threats and vulnerabilities for the system. Threats are identified via analysis of other views e.g. examination of system scenarios or "use cases"; based on inputs from requirements; through the involvement of domain or other experts; through brainstorming; and/or scrutiny of existing, similar systems.

Determine Risks. Determine likelihood of threats, their severity and overall risks.

Prioritize Threats. Based on Risk, or other heuristics, prioritize threats for mitigation.

Reconcile with Requirements. Systems will vary tremendously in the degree to which requirements articulate their Trust-relevant requirements, goals and needs. Sometimes there will be explicit requirements (often misleadingly labeled "non-functional requirements"). In most cases, Trust issues will be implicit, or couched in terms of regulatory or other constraints. Often the Architect will uncover situations which may lead to new or reformulated requirements related to Trust. As Threat model evolves and is completed, the results should be reconciled with system requirements: checked for coverage, adding new requirements when needed, etc.

5.3 Threat MK correspondence rules

Th-1: Each Threat must have a considered Risk.

Th-2: As Security model evolves, each Threat should be covered by one (or more) Mitigations (Measures or Mechanisms).

6 Risk assessment MK

Risk combines the probability of a threat occurring with the severity of the expected loss resulting from that threat, often in a form such as:[3]

$Risk = probability(ofThreat) \times severity(ofLoss)$

6.1 Risk assessment MK conventions

It is not the purpose of this viewpoint to establish Risk Assessment conventions. Frequently when Trust concerns are present, there is already a Risk Assessment and Management approach in place. Architects working on Trust should align with that approach. Such approaches will typically address:

- definition and classification of risks;
- risk identification, assessment and management methods;
- classification, levels of risk and severity (e.g., catastrophic, critical, marginal, negligible).

For example, see ISO 31000:2009 *Risk management — Principles and guidelines* and references cited therein.

For discussion of the integration of risk assessment into architecture methods see [2] and references cited therein.

6.2 Risk assessment MK correspondence rules

In the absence of further details on Risk assessment, there is only one correspondence rule **Th-1** (already specified above).

7 Security MK

> "Who has to trust whom for what to take this action?" [7]

[3] This traditional formulation, more generally, the limitations of probabilistic risk assessment for assessing and communicating the nature of risk, has been questioned (such as by [9]).

7.1 Security MK conventions

A *security model* captures the *security policies* the system will use, and how the system will enforce those security policies in the face of threats. The security model is expressed in terms of these fundamental constructs: *principals* (sometimes *subjects*), *actions* (sometimes *operations*) and *resources* (sometimes *assets* or *objects*).

Principals include people, organizations, system elements and other systems; principals are active agents which can perform actions upon resources. Principals may include stakeholders of the system. It is typically useful to classify the principals based on their roles and kinds of access they will have.

Resources include data in an information system, services and system capabilities, and shared system resources to be protected.

Actions express the ways in which principals interact with resources, including invocations of services, data flows, manual procedures, depending upon the type of system.

The principals, actions and resources will generally be drawn from other views of the system and categorized as such based on their roles in those views (see Operations on views), whereas the following constructs are "indigenous" to a trust view: *security policies*, *trust domains*, *measures and mechanisms*.

Security policies are sets of rules specifying, for each resource and each principal, what actions that principal can perform on that resource.

When more than one security policy is needed, the security model can be organized into one or more *trust domains*. A trust domain (or *information domain*, or simply *domain*) comprises a collection of principals and resources under a common security policy. Interactions across trust domains are often relevant; these should be scrutinized carefully [4].

Measures and mechanisms are the means by which resources are secured and systems are made trustworthy given the identified threats and considered risks. Examples include: data validation, user authentication, configuration management, cryptography, exception management, auditing and logging.

NOTE: *We use the term "measures and mechanisms" to reflect traditional usage. "Measure" tends to connote things of an operational nature (e.g., protect the perimeter, lock up input devices), whereas "mechanism" connotes system elements (e.g., firewalls, encryption). Since the Architect is dealing with* the whole system in its environment*, we want to encourage the widest possible interpretation by choosing that phrase.*

Data for the security model can be captured in tabular form, in a data store or modeling tool. However, a high-level graphic notation is suggested below that may be useful for sketching or documenting key cases.

There are no widespread, commonly used notations for threat or security models. [8] presents an informal graphical notation for an access control-based model (prin-

cipals, requests, guards and resources). [10] shows how to depict principals, actions and resources in UML, while primarily employing a tabular approach to documenting their Security perspective.

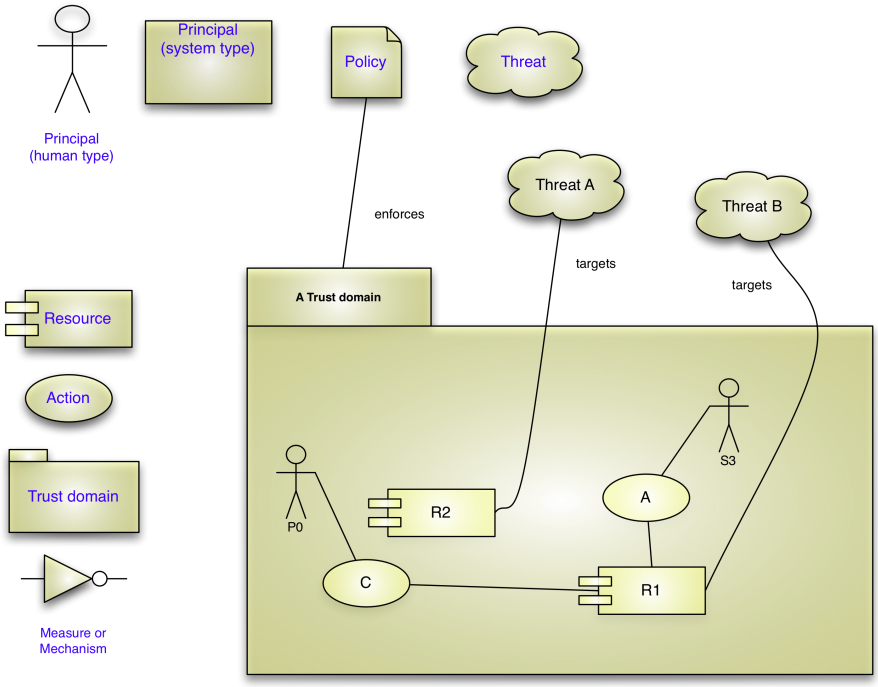

Figure 1: Graphical legend for trust constructs with example

7.2 Security MK operations

A future version will discuss the Avoid–Detect–Mitigate Pattern.

7.3 Security MK correspondence rules

As noted above, the Trust Viewpoint assumes it will be used in concert with some set of viewpoints, used to express other aspects of the architecture. The Security model should be linked to elements in those (undefined herein) views, such as follows:

Sc-1 Every principal should be linked to at least one element in another view.

Sc-2 Every resource should be linked to at least one element in another view.

Sc-3 Every action should be linked to at least one element in another view.

Possible sources of other viewpoints is discussed in the next section.

8 Operations on views

This section defines associated methods, activities, tactics, heuristics and guidelines which are applicable to create, interpret, analyze and use the views and constituent models produced with the Trust viewpoint.

NOTE: *The Trust viewpoint's models may be developed semi-independently, but must be integrated (i.e., made consistent) to satisfy this viewpoint; a useful trust view will also be integrated with other views of the system (see Correspondence rules).*

8.1 Construction

Mine/Analyze Requirements, other Views, for Threats. Take a look at other views for discovery of threats (e.g., Scenarios, Functional, Logical, Deployment, Context, Business).

There are various taxonomies (of threats, attacks and vulnerabilities, see [Hollingworth]). Taxonomies may be useful as starting points, or as checklists during analysis for the Architect creating a trust view.

There are best practices, heuristics and various folk wisdom from the security community, that may be helpful when starting a Trust view. A few example slogans: Keep it Simple (simple mechanisms, small number of mechanisms, one principalone mechanism); Defense-in-Depth; Isolation; Least privilege; Fail-safe; Replicate; Eschew security through obscurity; etc.

These will be discussed further in a future version.

Define Security Policies. Khare and Rifkin sketch three styles for approaching security policies [7]:

- principal-based;
- resource-based;
- action-based.

8.2 Analysis

There are several methods for analyzing the threat and security models defined above. Correspondence rules also assist the Architect to assess consistency with other views.

Taxonomies of threats, attacks and vulnerabilities (above) can help the Architect to assess coverage of the threat model. For deeper analysis, the threat model can be supplemented with representations such as attack graphs or with attack trees for some threats.

For security models, the literature offers numerous analysis techniques; two techniques with a long history are the flow model and the access control model (see [8]).

Trust Review. Convene independent, external reviews by domain experts.

Stakeholder Review of Mitigation. Convene reviews with stakeholders, such as via sample scenarios of proposed mitigations of selected threats.

9 Correspondence rules

Tr-1 Each threat in threat model should be linked to at least one measure or mechanism in security model to show the threat can be countered and identify the means and mechanisms used to counter it.

Utilizing other viewpoints. Additional CRs may relate to other viewpoints used in an architecture description. Additional possible sources is presented in the table 2 below.

10 Examples

A future version of the Trust viewpoint will include a worked example.

11 Notes

As part of the application of the Trust viewpoint to an architecture description, the architect must associate actual concerns with the actual stakeholders holding those concerns (in accordance with the Standard). The table below presents a sample assignment of which stakeholders may hold which concerns. It is only a sample.

Table 2: Possible viewpoint sources

Trust elements	Viewpoints
Potential principals	Functional, Logical
Trusted subjects, vulnerabilities	Information, Data
Resources, environment operational vulnerabilities, physical, infrastructure (3rd party) vulnerabilities	Deployment, Operational
User categories, Intended usage, potential threats, sources of threats	Context, Business,
Supplier vulnerabilities	Developer, Implementer

Table 3: Sample assignment of Stakeholders and Concerns

Concerns x Stakeholders	Developers	Users, Operators	Owners, Acquirers	Accreditors, Analysts
Confidentiality		✓		
Integrity		✓		
Availability		✓		
Threats		✓	✓	✓
Measures and Mechanisms	✓			✓
Policies	✓			✓
Accountability	✓		✓	✓

12 Sources

The Trust viewpoint described here has a long history. We first produced a security view along these lines in 1995 for the Army [4]. The present version is condensed from Security and Trust viewpoints discussed in 2001 [5]. The earliest work was

inspired by the Bell-LaPadula security model [1]. More recently, other security models have also been used [3].

References

Bell, D. and L. J. LaPadula. *Secure computer systems: unified exposition and Multics Interpretation*. Tech. rep. MTR–2297. Bedford, MA: The MITRE Corporation, 1976.

Bergomi, F. et al. "Beyond Traceability: Compared Approaches to Consistent Security Risk Assessments". In: *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*. Sept. 2013, pp. 814–820. DOI: 10.1109/ARES.2013.109.

Bidan, Christophe and Valérie Issarny. "Dealing with Multi-Policy Security in Large Open Distributed Systems". In: *Proceedings of the 5th European Symposium on Research in Computer Security*. Lecture Notes in Computer Science 1485. Belgium: Springer-Verlag, Sept. 1998, pp. 51–66.

Emery, David E., Rich Hilliard, and Timothy B. Rice. "Experiences Applying a Practical Architectural Method". In: *Reliable Software Technologies—Ada-Europe '96*. Ed. by Alfred Strohmeier. Lecture Notes in Computer Science 1088. Springer, 1996. URL: http://web.mit.edu/richh/www/writings/index.html#Experiences.

Hilliard, Rich. "Viewpoint Modeling". In: *First ICSE Workshop on Describing Software Architecture with UML*. Position paper. May 2001.

ISO/IEC/IEEE 42010, Systems and software engineering — Architecture description. Dec. 2011, pp. 1–46.

Khare, Rohit and Adam Rifkin. "Weaving a Web of Trust". In: *World Wide Web Journal* 2.3 (Summer 1997), pp. 77–112. URL: http://www.cs.caltech.edu/~adam/papers/trust.html.

Lampson, Butler W. "Computers at Risk". In: Washington: National Academy Press, 1991. Chap. Requirements and Technology for Computer Security, pp. 74–101.

Leveson, Nancy G. *Engineering a safer world: systems thinking applied to safety*. The MIT Press, 2011.

Rozanski, Nick and Eóin Woods. *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. Addison Wesley, 2005.

Contents