Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 157 (2006) 3–9
www.elsevier.com/locate/entcs

Why Trust is Bad for Security

Dieter Gollmann [1]

TU Hamburg-Harburg
Hamburg, Germany

Abstract

We investigate how the term 'trust' has been used, and re-defined, in computer security, covering Trusted Computing Platforms, Trust Management, Trusted Computing, and Trusted Code. We conclude that trust is a dangerous word to use as it has manifold and sometimes contradictory meanings. There is no immediate problem when trust is used in a specific research area to denote some concept of interest. Difficulties arise when interfacing between communities that use this word differently, and with the general public which is unlikely to associate a word like trust with any specific technical definition adopted in a field of research.

Keywords: Access control, security policies, trust, trusted code, trusted computing, trust management.

1 Introduction

IT security has been described as a fashion industry. The latest 'security technology' is bought because everybody else is doing the same (a.k.a. following best practice), but not because there has been a proper assessment of its actual merits, which admittedly can be difficult. Similar phenomena can be observed in security research. New terms arise, become the topic of research papers, and are included in calls for research proposals, without a coherent analysis of what these terms are supposed to stand for. IT also loves anthropomorphic explanations, such as sending emails or digitally signing documents, that can give a wrong idea about the service actually provided.

[1] Email: diego@tu-harburg.de

doi:10.1016/j.entcs.2005.09.044

Today, 'trust' is a prime example for such a fashionable anthropomorphic term. Trust is often regarded as an important precondition for the adoption of new technologies, and thus as a research problem needing urgent attention. Several research programs have listed trust as an important item in their calls for proposals. The goal is to provide technology that can be trusted by its users. When trust is defined, for example, as the user's willingness to risk time, money, and personal data on a website, we are primarily concerned with user psychology. In this context, trust is a natural term to use. Furthermore, this type of trust is an important goal indeed when designing and deploying IT systems. As we shall see, this interpretation does not necessarily match the various usages of trust in computer security.

We will not try to clarify the meaning of trust; too many different concepts are labelled with this word to make such an endeavour feasible. Instead, we will trace the history of 'trust' in computer security, identify some new research areas that express their goals as issues of trust, and try to convince the reader that it would be to the benefit of many of these research fields if they would state their agenda without reference to trust. Our investigations will focus on topics related to access control and to security policies.

2 Trusted Computing Base

When analyzing an IT system, one can identify components that have to work as expected for the system to meet its advertised purpose. Interpreting trust in a way that equates trust with expected behaviour, these components could be called trusted. In other words, a system is trusted (secure?) if there are no surprises when we use it.

In the early years of computer security, 'trusted' was used in this way, denoting everything one had to rely on for the system to remain secure. For example, the Trusted Computer System Evaluation Criteria [10] define a *Trusted Computing Base* (TCB) as

> the totality of protection mechanisms within a computer system – including hardware, firmware, and software – the combination of which is responsible for enforcing a security policy.

If a trusted component fails, security can be violated. We might want to have evidence that this cannot happen. In the 1980s, components that came with such evidence were called trustworthy. In contrast, trusted components are those that can hurt you. A similar use of the word trust can be found even earlier in the 1970s. In the Bell LaPadula model [1], trusted subjects are those subjects exempted from the mandatory security policies. To summarize our starting point of the history of 'trust' in computer security, trusted components

were those where you had to take on trust that they did not violate your security policies. For trustworthy components you had reasons to believe that this would be the case.

3 Trust Management

When the Internet was opened to general use, new paradigms developed in the way IT systems are used. It became obvious that policy administration amounts to more than setting access control lists by a single authority. Equally, policy enforcement amounts to more than checking an access control list stored with a protected resource. Policies can be encoded as certificates and policy enforcement may consult multiple policy decision points.

Trust management, as used in PolicyMaker [3] and KeyNote [2] was used as a term to distinguish a new and more general decentralized approach to access control from traditional methods that at one time where the only concepts around and might have been perceived as the essence of access control. (IT security sometimes suffers from confusing mechanisms with the services they provide.) We quote from [2]:

> Trust management, introduced in the PolicyMaker system, is a unified approach to specifying and interpreting security policies, credentials, and relationships.

The early papers on trust management had no need to define what they meant by trust as they did not propose new rationales for defining security policies, but just a set of new generic mechanisms. Unfortunately, later authors were lured by the term trust management to assume that these systems provided means for managing trust, inserting their own interpretation of trust in the process. The dangers inherent in the term trust management have been acknowledged by one of its creators [5]:

> Trust management is supposed to be an incredibly vague and provocative term invented by Matt Blaze. I don't know whether he intended it that way, but it comes natural to him

Decentralized access control is an interesting research area, covering topics such as policy languages, the design of enforcement mechanisms, or credential chain discovery services. In scenarios where there is no global entity defining the security policy but where we deal with an amalgamation of local policies defined by different entities, and where there are various ways of delegating (granting) access rights, we face a difficult task when defining the corresponding enforcement mechanisms. 'Precise' mechanisms that explicitly handle each possible case might become unwieldy, if not altogether impossible to design.

It is then an interesting question whether a generic algorithm would be acceptable that associates the credentials employed in the access control scheme with some weighting attribute, which might (in)conveniently be called trust, and bases fuzzy (or probabilistic) access control decisions on the combined trust value for the credentials provided (see e.g. [8] for an algorithm for combining such trust values). To deploy such a mechanism, we would have to justify that the attribute we happen to use does have the properties assumed by the decision algorithm, and policy administrators have to be guidance on assigning values for this attribute. In this context, the statement 'trust is whatever you want it to mean' does make some sense, but the research challenge is not so much the design of new algorithms for manipulating weights (trust) but to test those schemes in practice.

4 Trusted Computing

Concerns about the lack of security of Personal Computers connected to the Internet led in 1999 to the foundation of the Trusted Computing Platform Alliance (TCPA), which was later reconstituted as the Trusted Computing Group (TCG). The initial goal was to make the Web a safer place for surfers. The goals have changed and the website of the TCG now states its objectives as helping users protect their information assets (data, passwords, keys, etc.) from compromise due to external software attack and physical theft.

A core element in the security architecture developed by the TCG is a set of so-called roots of trust. The TCG Glossary [7] defines roots of trust as components that must always behave in the expected manner, because their misbehaviour cannot be detected. Such a root of trust would be a trusted component in the sense of section 2. The definition in the glossary continues with the statement:

> The complete set of Roots of Trust has at least the minimum set of functions to enable a description of the platform characteristics that affect the trustworthiness of the platform.

This makes a leap from trusted components to trustworthiness platforms, and destroys the careful distinction between trusted and trustworthy systems. With this interpretation of trust, we have reasons to believe that a trusted system will work as expected. Through attestation, there is even a mechanism whereby one can check that a remote system is configured as advertised.

On a technical note, the implementation of the roots of trust strongly relies on cryptographic mechanisms. The development of novel cryptographic mechanisms for trusted computing and the developments of applications that build on trusted computing modules are interesting research areas. Incidentally, one

can find discussions of cryptographic applications also outside Trusted Computing where trust stands for certain cryptographic keys, and where trust establishment mechanisms just amount to a key establishment protocols.

5 Trusted Code

The developments in the use of IT systems alluded to in section 3 have introduced code-based access control, as found in Java [6] or .NET [9], as an alternative to traditional identity-based schemes. In code-based access control, access privileges are assigned directly to code, not to users. It has become customary to refer to code running with many privileges as trusted code and to code running with very few privileges as untrusted code; for example, code restricted to a Java sandbox would be untrusted. Semi-trusted code might have a few more privileges but not the full set of privileges given to trusted code.

A flaw in code running with systems privileges might be exploited by an attacker to take over the victim's system. The same flaw in code that runs with limited privileges would have less serious implications. In this sense, trusted = privileged code is indeed a component that can hurt you, fitting the definition of trust initially adopted in computer security. However, calling code trusted may also insinuate that this is code you can trust, i.e. trustworthy code, which can easily lead to confusion.

In code-based access control, improving enforcement mechanisms that are currently based on a stack walk and the development of design patterns for the practical use of code-based access control are just two interesting current research challenges.

6 Policies and Mechanisms

The last remark above raises an important general issue. When deploying an access control system, we have the policy the system is supposed to enforce and the mechanisms that make the access control decisions. The mechanisms tend to be generic, the policies specific to a given application. The rationales for setting policies may differ and may sometimes relate to trust in a person or trust in a technology. However, there exist other rationales such as the need-to-know principle or contractual agreements between cooperating entities.

It is a potential problem when the language used to describe the mechanisms already suggests the rationale for setting a policy. In particular, when we have trusted code or trusted subjects, one might be led to assume that these entities have been given access rights because they are trusted in some

way. Maybe even worse, restricting someone's access rights could be read as an indication of a lack of trust (rather than a simple application of the need-to-know principle).

Consider, for example, a collaborative research project. Project participants will give members of partner organisations access to resources that are needed to work on the project, but not to their entire systems. Companies might even insist that there is a contractual agreement that specifies who will work on the project at partner organisations and the exact subset of project data these employees would be allowed to access. In all these decisions, the necessities of the project will have a dominant influence on the setting of the policy.

7 Conclusion

There are many reasons why trust is bad for security. First, security needs clarity and precision. Using a term like trust that has many different meanings (many more than explored in this paper) is unlikely to promote clarity and precision. As a point in case, TCBs do not guarantee trust, they ask for it; trust management does not manage trust but access control; using trust as a security attribute for access control mechanisms may give wrong ideas about the rationale for policies. We should thus take our cue from Occam's razor, quoted in one of its many variants:

> One should not increase, beyond what is necessary, the number of entities required to explain anything.

In this sense, it is generally a good idea to avoid overloaded terms, and trust is a much overloaded term indeed. There is, of course, no immediate problem when a specific research area decides to use trust for denoting some phenomenon of interest, presuming that is clear to everyone in the area what this term is supposed to mean. Problems may start when interacting with other research fields that use the same term, but with a different meaning; with due diligence it should be possible to be clear about the differences in the respective definitions so as not to fall into the trap of assuming everybody means the same as they are using the same word.

There are more serious problems when presenting any of the aspects of 'trusted' computing to a general audience. In such conversations, 'trusted' means what a reasonable person would understand it to mean, which is unlikely to match a specific technical definition. Conversely, the precise technical definition might actually mislead the public.

Returning to the starting point of this paper, users come to 'trust' complex technologies through experience, not because they understand it. In particular, many of the advanced access control schemes discussed today in the context of trust management are unlikely to have any direct impact on user confidence.

As a final note, our journey through the history of trust in computer security has touched on many interesting and novel research challenges. This leads to our last observation on trust. When facing new challenges it often takes time before the full nature of all its facets becomes clear. At such times, we need placeholders to keep the discussion going, but eventually those placeholders ought to be replaced by more precise terms. This situation is quite typical in security, take for example the reference to 'good' keys or the expression of authentication in terms of beliefs to be found in the BAN logic [4]. Another example is the $*$-property in the Bell LaPadula model [1] where the placeholder has survived till today. Trust is just the latest example for such a placeholder in times of changing security paradigms.

References

[1] D. E. Bell and L. J. LaPadula. Secure computer systems: Mathematical foundations and model. Technical Report M74-244, The MITRE Corporation, Bedford, MA, May 1973.

[2] Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D. Keromytis. The KeyNote Trust-Management System Version 2, September 1999. RFC 2704.

[3] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173, 1996.

[4] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *DEC Systems Research Center*, Report 39, revised February 22 1990.

[5] Joan Feigenbaum. Overview of the AT&T Labs trust-management project. In *Security Protocols, LNCS 1550*, pages 45–50. Springer Verlag, 1998.

[6] Li Gong, Mary Dageforde, and Gary W. Ellison. *Inside Java 2 Platform Security.* Addison-Wesley, Reading, MA, 2nd edition, 2003.

[7] Trusted Computing Group. *TCG Glossary*, July 2004. Revision 0.1.

[8] Audun Jøsang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(3):279–311, June 2001.

[9] Brian A. La Macchia, Sebastian Lange, Matthew Lyons, Rudi Martin, and Kevin T. Price. *.NET Framework Security.* Addison-Wesley Professional, Boston, MA, 2002.

[10] US Department of Defense. *DoD Trusted Computer System Evaluation Criteria*, 1985. DOD 5200.28-STD.