Veracity, Plausibility, and Reputation

Dieter Gollmann

diego@tu-harburg.de
Institute for Security in Distributed Applications
Hamburg University of Technology
21073 Hamburg, Germany

Abstract. The canonical IT security properties are geared towards infrastructure security. The task of an infrastructure security service is completed once data has been delivered to the application. When false data is submitted to the infrastructure, false data will be delivered securely. To secure an application, one thus may have to go beyond securing the infrastructure and in addition provide mechanisms for detecting false data. We propose *veracity* as a new security property relevant at the application level. We examine examples for veracity mechanisms from network management and conclude with a discussion of security in cyber-physical systems.

Keywords: Veracity, plausibility, reputation, security in cyber-physical systems

1 Introduction

The standard textbook introduction to IT security starts by defining *confidentiality*, *integrity*, and *availability* as the three pillars of security. These properties refer to the protection of data, be it data in storage (a main topic in operating system security) or data in transit (communications security). In both instances, we are looking at security properties of IT *infrastructures*. The job of a security mechanism is done once memory management or file management has securely delivered data to an application. The job of a security protocol is done once data has been securely delivered to the recipient.

ISO 7498-2 had added the notions of *authentication* and *non-repudiation* to confidentiality, integrity, and availability. Authentication and non-repudiation potentially take us away from a pure IT-centric view of security. Take, for example, the following definition from [10]:

> *Peer entity authentication* is provided for use at the establishment of, or at times during, the data transfer phase of a connection to confirm the identities of one or more of the entities connected.

This definition may suggest that a relationship between aspects of cyberspace, here connections, and entities in physical space, here the peers (people, machines)

connected, is being established. However, peer entity authentication establishes a link between connections and identities (names), i.e. it still describes a service within cyberspace. We have argued previously [9] that authentication services in general verify links between different aspects of the IT domain.

Similar observations apply to non-repudiation. The following definition is taken from ISO/IEC 10181-4.

> The goal of the *Non-repudiation* service is to collect, maintain, make available and validate irrefutable evidence concerning a claimed event or action in order to resolve disputes about the occurrence or non-occurrence of the event or action.

A sentence on the resolution of disputes strongly alludes to events outside the IT domain. However, cryptographic protocols cannot provide such a service on their own[1]. We would more appropriately refer to *unforgeable* evidence, and use non-repudiation for differentiating between the authentication services provided by symmetric key cryptography, where evidence can only be verified by a party in possession of the secret also used when creating the evidence, and the authentication services provided by public-key cryptography, where evidence can be verified with the help of a public verification key. Following this line of argument non-repudiation would be a security property within cyberspace[2].

The properties listed so far can be seen as security properties of an IT infrastructure. This state of affairs has a natural historical explanation. IT security has its root in the 1970s and 1980s when IT became a new infrastructure for data processing. Purists could justifiably complain that the term information technology was a misnomer as the services provided referred in the main to the storage and transmission of *data*, not to their interpretation. IT security services were provided primarily in *cyberspace*, and in cyberspace alone. When discussing such services, no statements are made about what happens after data have been delivered to the application consuming the data.

It is this latter aspect that this paper aims to explore. We begin by looking for security properties that reach into the application domain.

1.1 The Parkerian Hexad

Donn Parker's security framework [15] adds possession and utility to confidentiality, integrity, availability, and authenticity. *Possession* refers to control or ownership of data containers. This notion makes it possible to distinguish between violations of confidentiality, when information is disclosed to an adversary, and situations where the adversary has obtained a data container but cannot access the information within, e.g. because it is encrypted. *Utility* refers to usefulness (of data) for a given purpose. Possession and utility express relationships

[1] Consult, e.g., http://world.std.com/~cme/non-repudiation.htm

[2] As a historical footnote, [17] uses *veracity transaction receipts* as a synonym for unforgeable transaction receipts, i.e. for non repudiation. This paper will use the term veracity in a different meaning.

between cyberspace and applications in the physical world, as does integrity when interpreted in the meaning of *external consistency*.

> *External consistency*: The correspondence between the data object and the real world object it refers to [5].

1.2 Outlook

Security is moving to the application layer. This trend can be observed in computer security where the reference monitor is moving from the operating system into the browser and into web pages. This trend can be observed in communications security where security services are provided, for example, with the *http* protocol and in the layer above, such as in web services. The closer we move to the applications running on top of the IT infrastructure, the more urgently we need to ask whether the data provided by the infrastructure is actually fit for the intended purpose.

Utility addresses the issue whether the data received is presented in a useful format. We will pursue the question whether the data received truthfully captures the aspects of the physical world relevant for the application at hand. Section 2 will propose veracity as a new security property for cyber-physical systems. Section 3 gives three case studies that show how the veracity of assertions may be checked in network management. Section 4 deals briefly with plausibility and reputation. Section 5 discusses concepts from the business world intended for decreasing the likelihood of false assertions. Section 6 concludes with remarks on the security of cyber-physical systems.

2 Veracity

Assertions are statements about an aspect relevant in a given application domain. Assertions may, rightly or wrongly, be attributed to some entity in the application domain; authentication and non-repudiation verify the claimed origin of an assertion. Assertions may be true or false.

> *Veracity*: The property that an assertion truthfully reflects the aspect it makes a statement about.

Authentication and non-repudiation do not verify the veracity of assertions. Veracity is not a property guaranteed by any of the familiar IT infrastructure security services. Veracity is not a property that can be provided by familiar IT infrastructures. Veracity refers to aspects outside the IT infrastructure: the adversary is not an entity launching an attack in the infrastructure but an entity making false assertions. The data are already false when passed to the infrastructure.

Veracity is not an entirely new notion. In the field of databases veracity corresponds to external consistency. For mobile agent systems Borselius gives the following definition [2]:

> Veracity: The concept that an agent will not knowingly communicate false information.

Our definition is not restricted to assertions that are false on purpose but includes also assertions that are false by accident. This distinction does not matter for the application; the application would react to a false assertion in the same way in both cases. The distinction matters in the design of countermeasures. Borselius comments that veracity cannot be effectively enforced in mobile agent systems as *the required redundancy for such a [protection] system is likely to make the [mobile agent] system useless.*

Veracity can be achieved in two ways. We can protect the *sensor* making the observation reported in the assertion. The sensor must be tamper-resistant and it must not be possible to deceive the sensor by interfering with the observations it is expected to make. This approach may be applicable in scenarios where there is a sufficient degree of physical security.

Secondly, we can perform *consistency checks.* A consistency check may compare an assertion with a prediction made by a (local) *model* of the application. Alternatively, a consistency check may be performed on the facts reported in assertions made by several *witnesses.* Such schemes are only effective under the assumption that the adversary is unable to corrupt a sufficient number of assertions. This assumption moves us closer to research on reliability where mechanisms are designed on the basis that some but not all inputs are corrupted.

3 Veracity in Network Management

Although we have noted that there are no generic security mechanisms providing veracity services to applications running on top of familiar IT infrastructures, we can point to a few cases where there are checks on the veracity of assertions made in the context of network management.

3.1 DNS Rebinding

The Domain Name System (DNS) is a distributed directory service for binding host names to IP addresses. Most technical details of this service are not relevant for our discussion. The reader needs to be aware that *authoritative name servers* make assertions about such bindings for hosts in their zone, and that *resolving name servers* perform name resolution for their clients and cache bindings received with a time-to-live set by the authoritative name server. The resolving name server trusts the assertions received from the authoritative name server.

In DNS rebinding attacks the authoritative name server is telling lies. The adversary has its own domain with a corresponding authoritative name server and a host with a page containing a script that will issue requests to the victim's machine. The attack is launched via the browser of some unwitting user who has been lured to the adversary's web page.

When the user's browser resolves the adversary's host, the adversary's authoritative name server is consulted and binds the adversary's host first to the

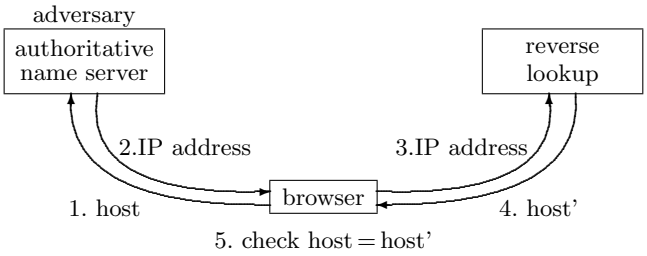

Fig. 1. Double checking the binding between host name an IP address.

true IP address (so that the attack page can be loaded) but then also to the IP address of the victim's machine. (Details about when and how this is done are again out of scope.) The same origin policy enforced by the user's browser permits scripts from the adversary's web page to connect back to the host they came from. When this host is bound to the victim's machine, requests sent to that machine will be passed on.

To defend against DNS rebinding, make it difficult for the adversary to get away with lies. Applying the same origin policy at the level of IP addresses fixes the original (true) IP address for the duration of its time-to-live in the cache [8], but this value is set by the authoritative name server. Pinning the original IP address for a period set by the browser delays the opportunity to substitute a false for a true assertion. The attack can become possible again if the browser can be induced to drop the pinning [12]. There are further problems when the same origin policy is relaxed and a script may connect to other sites authorized by the host the script originates from. Ultimately, we may resort to a consistency check and, similar to reverse DNS lookup, check that the host running on the given IP address is the one authorized by the policy (Fig. 1) [11].

3.2 Return Routability

In mobile IPv6 a node has a static home address in its home domain. The node is always addressable at this address. When a node has moved, messages sent to its home address will be forwarded by the home agent in an IPsec tunnel.

A node that has moved out of the home domain acquires a care-of address in its current domain. Correspondent nodes may be informed about the current care-of address as a performance optimization. This feature could be abused by a node claiming to be in a location it is not in and launching a denial-of-service attack by ordering a huge volume of data to be sent to that location. This attack cannot be prevented by authenticating the originator of the assertion (binding update) claiming that the node had moved to its new location. The assertion does come from the claimant.

The solution standardized in RFC 3775 and explained in [1] performs the following consistency check. The correspondent node sends two test messages to

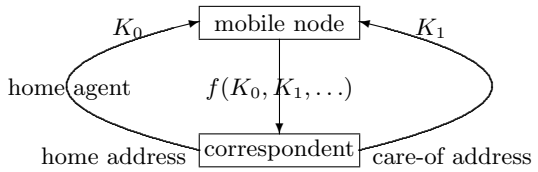

Fig. 2. Return routability: check that two challenges have arrived at the same location.

the claimant, one to its home address, from where it would be forwarded to the node by its home agent, the other directly to the care-of address given. Both test messages contain a key transmitted in the clear. A node in the claimed location will receive both keys, which are required for creating a valid response (Fig. 2). When the correspondent node receives a valid response it concludes that the mobile node is in the location asserted (return routability).

This protocol relies on the assumption that the adversary is unable to obtain both keys. How this is achieved, e.g. by using IPsec between correspondent node and home agent, is outside the scope of the protocol. The security service provided could be denoted as *location authentication* [9] when we focus on the binding between home address and care-of address. Alternatively, we may view this mechanism as a veracity check on an assertion about a current care-of address.

3.3 Secure Network Coordinates

Network coordinate systems in the Internet, such as Vivaldi [7], assign synthetic coordinates to nodes. Coordinates are computed based on round trip times measured between some of the nodes. The coordinates are then used for predicting communication latency between other nodes. Such schemes can be subverted by parties making false assertions about round trip times. Such schemes can be secured by establishing the veracity of assertions. Proposed security mechanisms rely on:

- trusted surveyor nodes (witnesses) [13],
- models that capture some aspects of the physical domain, e.g. a triangle inequality test for round trip times as in PIC [6],
- a set of untrusted verification nodes that vote on the veracity of an assertion, as in Veracity [16]; each verification node has its own local model to decide whether to accept an assertion.

Models are used for detecting deviations from expected or from possible behaviour. An assertion making an impossible claim is clearly false. An assertion making an unexpected claim may be true. A false assertion may still report an expected value. We thus must not place too much reliance on the prediction of a single model. Security can be strengthened by employing several models. Assessing the strength of a combination of models requires some understanding of their dependencies.

4 Plausibility and Reputation

In contrast to the first two case studies where a single node reached a binary decision on the veracity of an assertion based on checks it could perform on its own, schemes in the third case study consider inputs from several witnesses and arrive at a decision about the *plausibility* of an assertion received.

> An assertion is *plausible* if it does not deviate too far from the values estimated by the models used.

Schemes that rely on witnesses can be undermined by witnesses that make false statements. Schemes can be designed so that they can handle a certain proportion of witnesses colluding in an attack. Security proofs will be based on the assumption that the adversary can compromise only a limited number of witnesses.

The *constrained-collusion Byzantine failure model* [4] may serve as an example. A fraction f, $0 \leq f < 1$, of the N nodes in a system may be faulty (compromised). Faulty nodes can behave arbitrarily. Faulty nodes can be partitioned into disjoint coalitions bounded in size by cN, $1/N \leq c \leq f$. For $c < f$ there are multiple independent coalitions; for $c = f$ all faulty nodes may collude.

Witness selection has an impact on how easy it is for the adversary to predict which witnesses to compromise so that a collusion attack is effective. Veracity, for example, uses a deterministic selection scheme based on hash chains [16]. For security arguments based on statistical independence assumptions, the case needs to be made that such assumptions can be substantiated in practice.

Schemes that rely on witnesses may be augmented by features that try to detect and exclude witnesses making false statements. *Reputation systems* employ witnesses on witnesses, i.e. entities that make statements (reputation ratings) about the veracity of statements made by other witnesses. CONFIDANT, for example, applies this approach to source routing in mobile ad-hoc networks [3]. A security analysis of a reputation system must now in turn consider the impact of false reputation ratings.

5 Fostering Veracity in Business Processes

Commercial applications are a further area to turn to for cues on how veracity may be checked and maintained. The Clark-Wilson model captures established business practices and proposes *separation of duties* as a design principle *to control fraud and error*, i.e. for maintaining external consistency [5]. A process is split into parts that must be executed by different entities. We may view the different entities involved as witnesses with respect to the actions of their predecessors. A related design principle requires that a quorum of entities approves a transaction before it can go ahead.

These principles are similar to those deployed for ensuring veracity in network management. Multiple entities are involved so that a collusion by a small subset can be detected.

6 Securing Cyber-physical Systems

The "applications" in Section 3 had been network management functions. These applications were still within the IT domain but we can generalize the fundamental principles observed to securing cyber-physical systems.

Consider an applications with observations and effects in the physical world, relying on services provided in "cyberspace" by an IT infrastructure. An adversary can attack the application by feeding it with data causing undesired effects. Stuxnet may serve as an illustrative example [14]. Securing the IT infrastructure protects against an adversary who can manipulate data as it is processed in the infrastructure, but not against an attacker who can feed manipulated data to the infrastructure, possibly by manipulating the sensors that supply the data. To quote from [14]:

> *Manipulations of a controller have less to do with the confidentiality, integrity, and availability of information and more to do with the performance and output of a physical production process.*

Defences outside cyberspace may focus on the sensors and controllers that provide the interface between physical space and cyberspace. Defensive measures include:

- physically tamper-resistant sensors and controllers,
- attestation of the software configurations of programmable devices, maybe on the basis of Trusted Platform Modules, together with secure update procedures,
- controlling the environment of sensors so that an adversary cannot induce a truthful reporting of distorted facts that would promote the adversary's objectives.

Defences that secure devices can be implemented more easily in closed, controlled environments. When such defences are not feasible, countermeasures can take the form of plausibility checks on inputs received and plausibility checks on actions performed. Plausibility checks rely on models of the physical system that capture the expected behaviour of the application. In the most simple case, we could model that two sensors observe the same quantity so that their readings should be the same within the bounds of measurement errors. We may further model the relationship between different quantities to check the consistency of a collection of observations. We may also use models to check the consistency between predicted and observed behaviour.

Security analysis is performed in a threat model where not all components in the system are trusted. Concepts from Byzantine fault tolerance become relevant in this context. The efficiency of security mechanisms may improve when suitable independence assumptions can be made. It must, however, be noted that provably secure systems are ever so often broken by violating some of the assumptions of the security proof; independence assumptions would appear to present an attractive target.

References

1. Aura, T., Roe, M., Arkko, J.: Security of Internet location management. In: Proceedings of the 18th Annual Computer Security Applications Conference. pp. 78–87 (December 2002)
2. Borselius, N.: Mobile agent security. Electronics & Communication Engineering Journal 14(5), 211–218 (2002)
3. Buchegger, S., Boudec, J.Y.L.: Performance analysis of the CONFIDANT protocol. In: MobiHoc. pp. 226–236 (2002)
4. Castro, M., Druschel, P., Ganesh, A.J., Rowstron, A.I.T., Wallach, D.S.: Secure routing for structured peer-to-peer overlay networks. In: OSDI (2002)
5. Clark, D.R., Wilson, D.R.: A comparison of commercial and military computer security policies. In: Proceedings of the 1987 IEEE Symposium on Security and Privacy. pp. 184–194 (1987)
6. Costa, M., Castro, M., Rowstron, A.I.T., Key, P.B.: PIC: Practical internet coordinates for distance estimation. In: ICDCS. pp. 178–187 (2004)
7. Dabek, F., Cox, R., Kaashoek, M.F., Morris, R.: Vivaldi: a decentralized network coordinate system. In: ACM SIGCOMM. pp. 15–26 (2004)
8. Dean, D., Felten, E.W., Wallach, D.S.: Java security: from HotJava to Netscape and beyond. In: Proceedings of the 1996 IEEE Symposium on Security and Privacy. pp. 190–200 (1996)
9. Gollmann, D.: Authentication by correspondence. IEEE Journal on Selected Areas in Communications 21(1), 88–95 (January 2003)
10. International Organisation for Standardization: Basic Reference Model for Open Systems Interconnection (OSI) Part 2: Security Architecture. Genève, Switzerland (1989)
11. Jackson, C., Barth, A., Bortz, A., Shao, W., Boneh, D.: Protecting browsers from DNS rebinding attacks. In: Proceedings of the 14th ACM Conference on Computer and Communications Security. pp. 421–431 (2007)
12. Johns, M.: (Somewhat) breaking the same-origin policy by undermining DNS pinning. Posting to the Bug Traq mailing list (August 2006), `http://www.securityfocus.com/archive/107/443429/30/180/threaded`
13. Kaafar, M.A., Mathy, L., Barakat, C., Salamatian, K., Turletti, T., Dabbous, W.: Securing internet coordinate embedding systems. In: ACM SIGCOMM (August 2007)
14. Langner, R.: Stuxnet: Dissecting a cyberwarfare weapon. IEEE Security & Privacy 3(9), 49–51 (2011)
15. Parker, D.B.: Toward a new framework for information security. In: Bosworth, S., Kabay, M. (eds.) Computer Security Handbook, chap. 5. John Wiley & Sons, fourth edn. (2002)
16. Sherr, M., Blaze, M., Loo, B.T.: Veracity: Practical secure network coordinates via vote-based agreements. In: USENIX Annual Technical Conference (USENIX-ATC). USENIX (June 2009)
17. Simmons, G.J., Purdy, G.B.: Zero-knowledge proofs of identity and veracity of transaction receipts. In: Günther, C. (ed.) Advances in Cryptology - Eurocrypt'88, LNCS 330. pp. 35–43. Springer Verlag (1988)