SECURITY ANALYSIS USING NETWORK PROTOCOL

[1]Dr.V.Sangeetha
Assistant Professor
Information Technology
Sankara college of science and commerce, Coimbatore, India

Abstract: Network security protocols, such as key-exchange and key-management protocols, are notoriously difficult to design and debug. Anomalies and shortcomings have been discovered in standards and proposed standards for a wide range of protocols, including public-key and Diffie-Hellman–based variants of Kerberos, SSL/TLS, and the 802.11i (Wi-Fi2) wireless authentication protocols. Although many of these protocols may seem relatively simple, security protocols must achieve their goals when an arbitrary number of sessions are executed concurrently, and an attacker may use information provided by one session to compromise the security of another.

***IndexTerms* – Network security, key management protocols, wireless authentication.**

I. INTRODUCTION

Protocols that enable secure communication over an untrusted network constitute an im-portant part of the current computing infrastructure. Common examples of such protocols are SSL , TLS , Kerberos , and the IPSec and IEEE 802.11i protocol suites. SSL and TLS are used by internet browsers and web servers to allow secure transac-tions in applications like online banking. The IPSec protocol suite provides confidentiality and integrity at the IP layer and is widely used to secure corporate VPNs. IEEE 802.11i provides data protection and integrity in wireless local area networks, while Kerberos is used for network authentication.

The design and security analysis of such network protocols presents a difficult prob-lem. In several instances, serious security vulnerabilities were uncovered in protocols many years after they were first published or deployed . While some of these attacks rely on subtle properties of cryptographic primitives, a large fraction can be traced to intricacies in designing protocols that are robust in a concurrent execution setting. To further elaborate this point, let us consider the concrete example of the SSL protocol. In SSL, a client typically sets up a key with a web server. That key is then used to protect all data exchanged between them. A single client can simultaneously engage in sessions with multiple servers and a single server concurrently serves many clients. Let us consider a scenario in which all network traffic is under the control of the adversary. In addition, the adversary may also control some of the clients and servers. The protocol should guarante certain security properties for honest agents even in such an adversarial environment.

.

II. PROTOCOL DERIVATION SYSTEM

Researchers and practitioners working in the field of protocol security recognize that common authentication and key exchange protocols are built using an accepted set of stan-dard concepts. The common building blocks include Diffie-Hellman key exchange, to avoid replay, certificates from an accepted authority to validate public keys, and en-crypted or signed messages that can only be created or read by identifiable parties. An informal practice of presenting protocols incrementally, starting from simple components and extending them by features and functions,

III. DERIVATION OF THE STS FAMILY

The basic components, and the composition, refinement and transformation operations used in deriving the STS family of key exchange protocols. The components and operations are presented tersely, with additional intuition and explanation given where they are used.

In informally describing the derivation system, we use a standard messages-and-arrows notation for protocol steps. Experience suggests that this simple notation is useful for conveying some of the central ideas. However, the reader should bear in mind that, in addition, a protocol involves initial conditions, communication steps, and internal actions. When we derive a protocol, the derivation step may act on the initial conditions, network messages.

IV.PROTOCOLCOMPOSITIONLOGIC

Protocol Composition Logic (PCL) is a logic for proving security properties of network protocols. The logic is designed around a process calculus with actions for each protocol step. Protocol actions are annotated with assertions in a manner resembling dynamic logic for sequential imperative programs. The semantics of our logic is based on sets of traces of protocol executions, following the standard symbolic model of protocol execution and attack. Security proofs involve local reasoning about properties guaranteed by individual actions and global reasoning about actions of honest principals who faithfully follow the protocol. One central idea is that assertions associated with an action will hold in any protocol execution that contains this action. This observation gives us the power to reason about all possible runs of the protocol without explicitly reasoning

4.1compositional proof method

A method for reasoning about compound protocols from their parts. In general terms, we address two basic problems in compositional security. The first may be called *additive combination* – we wish to combine protocol components in a way that accumulates security properties. For example, we may wish to combine a basic key exchange protocol with an authentication mechanism to produce a protocol for authenti-cated key exchange. The second basic problem is ensuring *nondestructive combination.* If two mechanisms are combined, each serving a separate purpose, then it is important to be sure that neither one degrades the security properties of the other. For example, if we add an alternative mode of operation to a protocol, then some party may initiate a session in one mode and simultaneously respond to another session in another mode, using the same public key or long-term key in both. Unless the modes are designed not to interfere, there may be an attack on the multi-mode protocol that would not arise if only one mode were possible. An interesting illustration of the significant of nondestructive combination.

4.2 Abstraction and Refinement Methodology

Protocol templates can also be used to formalize the informal practice of protocol design by combining different mechanisms. The key obser-vation is that if a concrete protocol is an instantiation of two different protocol templates, each instantiation respecting the assumed invariants associated with the template, then the concrete protocol has the security properties of both templates. A protocol template can be instantiated to multiple protocols. Security proofs of instances of a template follow from the proof of the template plus a (usually much simpler) proof that the instances satisfy the assumed hypotheses. A refinement operation, when applied to a protocol, adds an additional security property while preserving the original properties. Examples of refinement operations considered include replacing signatures by encrypted signatures to provide identity protection and replacing fresh Diffie-Hellman exponentials by a pair consisting of a stale exponential and a fresh nonce, thereby enabling reuse of exponentials and hence greater computational efficiency. The methodology for combining protocol templates, described in Section 4.2.2, provides a way to formally reason about a broad class of refinements including the two just mentioned. Below we illustrate the general method by examining the identity protection refinement in some detail.

IV. CONCLUSION

Our main contribution is PCL—a logic for proving security properties of network protocols. Security proofs in PCL are relatively short and intuitive and scale to protocols of practical interest. Two central results for this logic are a composition theorem and a computational soundness theorem. The composition theorem allows proofs of com-plex protocols to be built up from proofs of their constituent sub-protocols. It is formulated and proved by adapting ideas from the assume-guarantee paradigm for reasoning about distributed systems. The computational soundness theorem guarantees that, for a class of security properties and protocols, axiomatic proofs in a fragment of PCL carry the same meaning as hand-proofs done by cryptographers. The soundness proof uses standard proof techniques from cryptography, in particular, complexity-theoretic reductions. In addition, we have developed an abstraction-refinement method for proving protocol properties in an abstract template form using a higher-order extension of PCL. PCL has been applied to the IEEE 802.11i protocol suite (which includes TLS as a component) [60] and to the IETF GDOI protocol for secure group communication

.

REFERENCES

[1] IEEE P802.11i/D10.0. Medium Access Control (MAC) security enhancements, amendment 6 to IEEE Standard for local and metropolitan area networks part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications., April 2004.

[2] IEEE P802.16e/D10.0. IEEE Standard for local and metropolitan area networks. part 16: Air interface for fixed and mobile broadband wireless access systems. amend-ment for physical and medium access control layers for combined fixed and mobile operation in licensed bands., August 2005.

[3] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *28th ACM Symposium on Principles of Programming Languages*, pages 104–115, 2001.

[4] M. Abadi and A. Gordon. A calculus for cryptographic protocols: the spi calculus. *Information and Computation*, 148(1):1–70, 1999. Expanded version available as SRC Research Report 149 (January 1998).

[5] M. Abadi and A. D. Gordon. A bisimulation method for cryptographic protocol. In *Proc. ESOP 98*, Lecture notes in Computer Science. Springer, 1998.

[6] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: the spi calcu-lus. *Information and Computation*, 143:1–70, 1999. Expanded version available as SRC Research Report 149 (January 1998).