

Towards Validating Security Protocol Deployment in the Wild*

Luca Compagna
SAP Labs France
805, Avenue du Dr Maurice Donat
06250 Mougins
France
luca.compagna@sap.com

Ulrich Flegel
SAP Research CEC Karlsruhe
Vincenz-Prießnitz-Str. 1
76131 Karlsruhe,
Germany
ulrich.flegel@sap.com

Volkmar Lotz
SAP Labs France
805, Avenue du Dr Maurice Donat
06250 Mougins
France
volkmar.lotz@sap.com

Abstract

As computing technology becomes increasingly pervasive and interconnected, mobility leads to shorter-lasting relationships between end-points with many different security requirements. Also the rapid development of new service landscapes calls for standardized, yet highly flexible security protocols. It has been demonstrated that the increasing number of application contexts of these highly flexible security protocols opens vulnerabilities emerging from the difficulty of assessing the impact of the selected protocol options on the actual security of the relationship established using the protocol.

This contribution clearly identifies the underlying problem more generally and establishes the need for run-time on-the-fly verification of security protocol instances considering the actual choice of options and environment assumptions. Extending security protocol verification from the design-time realm into deployment and even run-time generates various challenges. This paper identifies these new challenges and proposes directions for research on solutions.

*This work was partially supported by the following projects: FP7-ICT-2007-1 project no. 216471, "AVANTSSAR: Automated Validation of Trust and Security of Service-oriented Architectures" (www.avantssar.eu); German Federal Ministry of Economy and Technology promotional reference 01MQ07012, "THESEUS/TEXO" (<http://theseus-programm.de>). The authors take the responsibility for their contribution.

1 Motivation

In the past, there has always been a strong case for formal validation of security protocols. Formal analyses allowed to discover flaws that have previously gone unnoticed, even for protocols that had been deployed for years and had been subject to thorough expert analysis. A whole discipline of formal protocol analysis has emerged from these discoveries, not only because it turned out to be a rewarding application domain for formal methods research, but also for its practical relevance and its strong contribution to the notoriously difficult task of security analysis. Meanwhile, we have techniques at our disposal that allow to analyse large protocols, i.e., those with a large number of protocol steps, messages and message fields. To mention but a few, AVISPA [1], ProVerif [4], and Scyther [5] have been successfully applied to complex and industrial relevant security protocol specifications like those proposed by ITU and IETF.

We feel a need to reinforce the importance of formal protocol validation in modern information and communication technology environments, while at the same time motivating an extended usage scenario required by these environments. In particular, we want to promote the validation of protocol *deployment* instead of protocol *definition*, with the major implication that validation needs to be done on-the-fly, sometimes even at run-time. By protocol deployment, we mean the actual instance of a protocol specification implemented by an entity (e.g., a service provider or a mobile device) with its specific selection of options and assump-

tions about the execution and application context. While protocol validation has so far focused on single reference scenarios (here called a protocol definition), an entity that frequently and briefly interacts with a multitude of services has to cope with the full variety of deployment options and environment constraints with their respective impact on the validity of security properties. Recent work has shown how sensitive protocols are to their respective environment, even if the reference scenario is proven to be secure [2].

Consider the following example: Upon roaming, a mobile device will decouple from the initial static local environment and re-engage with a closer and reachable environment, this process being iterated many times. Every environment may offer various services, each with its distinct requirements for and assumptions on security. As a result, a given mobile device will establish and maintain relationships with many service providers, each of them offering their own options and relying on their own assumptions. Such a setting requires a small number of very versatile standardized security protocols, which during runtime adapt to the requirements of the offered service, as well as to the specifics of the end point platforms.

Verifying all potential protocol deployment scenarios in advance is not an option, due to the multitude of potential scenarios (cf. the options and different profiles of the SAML2.0 set of standards as example) and the uncertainty about the assumptions and requirements made. We rather propose to aim at deriving the respective scenario upon the encounter with a peer entity (or set of entities), and to use efficient, fully automated techniques to validate the desired security properties on the fly. Existing technology for security protocol validation provides a solid starting point. In this contribution we explore the conceivable extensions that will be necessary.

2 Position statement

Modern information and communication technology environments want to be increasingly open to the entire worldwide community. In these settings, security protocols are required to offer the degree of flexibility and options needed to meet the multitude of requirements emerging from different actors that want to access these environments. The more the protocol will be open and flexible, the more the actors will be able to interoperate by using it, but on the other hand the chances of unpredicted and undesired protocol deployments violating critical security desiderata increase significantly.

The work discussed in [2] is illustrative in this context. The SAML 2.0 Single Sign-On (SSO) protocol specification [8] from OASIS has been devised to provide a standardized, open, interoperable SSO solution applicable in a multitude of environments and situations. Not surprisingly, it is

widely considered a key enabler for pervasive computing, service-oriented computing, SaaS and other similar emerging trends. Clients, service providers (SPs) and identity providers (IdPs) can choose from a significant number of protocols, message, and binding options to set-up their SSO services and to communicate with each other. Google Apps offers a SAML-based SSO IdP service that, once deployed at its customer organizations, allows the clients of these organizations a direct and transparent access to the popular web-based Google applications like Gmail or Google Calendar. The first version of this SAML-based SSO service for Google Apps has been released by Google in February 2007; formal analysis by means of SATMC, a state-of-the-art model checker for security protocols [3], discovered a serious security flaw in June 2008. The attack, meanwhile fixed, allowed a dishonest service provider to access Google Apps under the identity of an unaware user. The problem was due to the absence of a few but critical fields in the authentication assertion generated by the IdP. The reasons for not having used those fields are unknown, however, the point we want to make is independent of them: the chosen deployment options do have an impact on security, and the fact that the choice is made independently by the respective provider bears a risk of insecure protocol instances being deployed.

Formal protocol validation becomes thus even more important than in the past to ensure on-the-fly (i.e., at the time before actual protocol executions) that the protocol specification options are properly selected to comply with the requirements imposed by the actors that want to run the protocol and to guarantee the overall protocol security desiderata. As a guiding vision we imagine that during the pairing of the end-points and after choosing the instance parameters and variants for a security protocol, a verification of the security of the protocol instance is executed in the background. It is obvious, that run-time verification could not see the actor as responsible for providing guiding input, rather actor interaction must be minimal. While this poses additional challenges, restricting the search space to the run-time instance of the protocol might enable such a technology to run on mobile devices with limited resources.

Though existing validation methodologies have been proven to be successful on traditional security protocols, modern environments and emerging trends pose new challenges that need to be tackled:

Protocol specification vs protocol definition: It is not anymore a single security protocol that has to be considered, but a family of security protocols defined by the protocol specification, enumerating the multitude of options (e.g., the use of fields and protocol profiles) and assumptions (e.g., on properties of communication channels used).

Preferences, policies, constraints emerging by the actors:

Each actor that wants to employ one of the security protocols defined by the protocol specification comes with its own preferences, policies, constraints that narrow down the freedom in the selection of options for the given execution environment.

Attacker threat model: Traditional approaches to validate security protocols assume the popular Dolev-Yao threat model [6] (DY in brief) which considers an omnipotent malicious principal controlling the whole network traffic. The usual justification is that a protocol secure against DY certainly is secure against a less powerful, perhaps more realistic attacker. This justification does not work for security protocols analysis in modern environments. Here the attacker threat model has to comply, as faithful as possible, with the protocol assumptions and the environment in which the deployed protocol will be run, without any overwhelming capability. For instance, if a certain communication channel is assumed to be unreadable for the attacker, an attacker unable to access that channel at all would satisfy this requirement, but an attacker unable to read it would satisfy that requirement more faithfully. Validating a protocol specification against a too powerful attacker may result in deploying a protocol that is far too demanding and costly than the one that could have been soundly deployed, like shooting a bird with a cannon ball.

On-the-fly validation: Formal validation has to be undertaken on-the-fly, before executing the chosen security protocol instance. This protocol instance has to comply with the actor preferences still achieving its critical security desiderata. It remains an open point where and how to undertake this formal validation. Likely, each actor that desires to initiate an interaction with others using the protocol specification wants to locally perform the validation step. Alternatively, a trusted-third party might be introduced to perform an overall validation. In both the cases the set of preferences, policies, and constraints of each actor has to be provided as input to the validation entity and trustworthiness of this information needs to be established.

The intuition is hereafter explained. Let \mathcal{P} be a traditional *security protocol* supposed to achieve certain *security desiderata* G (e.g., secrecy of data, authentication, non-repudiation, etc). \mathcal{P} can be defined as a set of *role behaviours* parametric on certain data parameters, i.e., $\mathcal{P} = \{R_1(P_{1,1}, \dots, P_{1,k_1}), \dots, R_n(P_{n,1}, \dots, P_{n,k_n})\}$. For instance, it is easy to define the well-known Needham-Schroeder Public Key protocol [7] (NSPK in brief) in terms of the behaviour of its roles initiator and responder where the role parameters range over actor identifiers, actor

keys, etc. Roles are played by actors. An *actor session* $A(R\sigma_{A,\#}, \mathcal{P}, \#)$ defines the actor A playing the role R ($\sigma_{A,\#}$ properly instantiates R 's parameters) of protocol \mathcal{P} in session $\#$ (sessions allow the same actor to play the same protocol role more than once). An interleaving of actor sessions where all the protocol roles are played at least once defines a protocol *scenario*. For instance, a protocol scenario for NSPK can be

$$\|\{ \text{alice}(\text{initiator}(\dots)\sigma_{\text{alice},1}, \text{NSPK}, 1), \\ \text{charlie}(\text{responder}(\dots)\sigma_{\text{charlie},1}, \text{NSPK}, 1), \\ \text{bob}(\text{initiator}(\dots)\sigma_{\text{bob},2}, \text{NSPK}, 2), \\ \text{alice}(\text{responder}(\dots)\sigma_{\text{alice},2}, \text{NSPK}, 2) \}$$

where *alice* and *charlie* engage in session 1 of the protocol, while *bob* and *alice* communicate in session 2.

Existing formal validation approaches are able to establish whether a protocol scenario S interleaved with the attacker threat model I (normally the Dolev-Yao [6]) achieves the protocol desiderata G , in symbols $S, I \models G$. When G is limited to secrecy and authentication, certain approaches can even answer whether the security desiderata are achieved by the protocol applied in any scenario, i.e., $\forall S, I \models G$.

Security protocols for modern environments are defined by means of protocol specifications and are employed by actors that have their own preferences, policies and constraints. A protocol specification \mathcal{PS} defines a set of role specifications $\{R_1, \dots, R_n\}$ and offers a multitude of options (e.g., optional message fields) whose configurations produce a family of likely similar, but surely not identical security protocols $\{\mathcal{P}^1, \dots, \mathcal{P}^m\}$ each one defining specific role behaviours for R_1, \dots, R_n . Let $rs(\mathcal{PS}) = \{R_1, \dots, R_n\}$ and $p(\mathcal{PS}) = \{\mathcal{P}^1, \dots, \mathcal{P}^m\}$. Moreover, let $rb(R_j, \mathcal{P}^i)$ be the role behaviour associated to the role specification $R_j \in rs(\mathcal{PS})$ of protocol $\mathcal{P}^i \in p(\mathcal{PS})$. The preferences, policies and constraints of an actor A are captured by the formula $ap(A)$.¹ Agent A that wants to play the role specification $R \in rs(\mathcal{PS})$ in a session $\#$ can deploy any role behaviour of R that satisfies $ap(A)$. Let $canplay(A, R, \mathcal{PS}, ap(A))$ be the set of those protocols $\{\mathcal{P}^{i_1}, \dots, \mathcal{P}^{i_k}\} \subseteq p(\mathcal{PS})$ such that $rb(R, \mathcal{P}^{i_j})$ complies with $ap(A)$ ($j = 1, \dots, k$). An *actor session specification* $A[R, \mathcal{PS}, \#]$ denotes actor A that wants to play the role specification $R \in rs(\mathcal{PS})$ in session $\#$. A set of actor session specifications defines a protocol *scenario specification* provided that all the role specifications in $rs(\mathcal{PS})$ are played at least once.

Given a protocol specification \mathcal{PS} and its security desiderata G , a scenario specification SS , the actor preferences $ap(A)$ for each actor A , and an attacker threat model I

¹Clearly a proper language and formalism needs to be introduced to express $ap(A)$. State-of-the-art formal languages for policy specification might be helpful here.

properly tailored on and complying with the protocol specification and actor preferences, the validation problem we are interested to solve is thus:

$$\forall A[R, \mathcal{PS}, \#] \in SS. \forall \mathcal{P}^i \in \bigcap_{A,R} \text{canplay}(A, R, \mathcal{PS}, \text{ap}(A)). \\ \|\{A(\text{rb}(R, \mathcal{P}^i)\sigma_{A,\#}, \mathcal{P}^i, \#)\}, I \models G \quad (1)$$

More precisely, we want to solve multiple validation problems, one for each suitable protocol deployment complying with the respective actors' preferences and with the given scenario specification. Any violation of the validation problem $\|\{A(\text{rb}(R, \mathcal{P}^i)\sigma_{A,\#}, \mathcal{P}^i, \#)\}, I \models G$ indicates that that protocol deployment should be avoided (e.g., the actors can make their preferences more strict) as it is vulnerable to a security attack. Notice that $\bigcap_{A,R} \text{canplay}(A, R, \mathcal{PS}, \text{ap}(A))$ might result in an empty set indicating that the preferences expressed by the various actors make them unable to deploy any of the protocols in $p(\mathcal{PS})$. Last but not least, it is worth mentioning that if the actor preferences, policies and constraints comprise requirements on the behaviours of other actors, the operator *canplay* should be changed to keep this into account and to consider the union of the actor preferences rather than the preference of the single actor.

As mentioned above, one additional challenge concerns where to check the validity of (1). Each actor might want to perform this check locally, but there are situations in which this might be infeasible (e.g., when the actor has not enough computational capabilities). A trusted-third party can be used in those cases. Clearly the trustworthiness of input for the validation problem has to be established to make the validation itself meaningful.

3 Research Directions

Our proposition is assessing the impact of a selected scenario and environment constraints on the security of a protocol. Hence, languages are required that express these preferences and restrictions. While existing policy languages may be useful here, it remains to be investigated if they provide enough intuition for the modeler and can express that information sufficient for computing, for instance, the *canplay* set (see Sect. 2). We expect that *canplay* may generate huge result sets, such that traditional validation techniques may run into problems. Thus, it is worth considering combining the validation techniques with policy reasoning techniques to narrow down to a manageable focus for validation.

Considering the feasibility of on-the-fly validation it is necessary to quantify the complexity of such a validation run and how to cope with a mixture of trustworthy inputs from the local platform and with the input from the remote end-point. Considering very limited devices, it could be

necessary to have the surrounding infrastructure carrying out the validation. Such a solution requires strong guarantees on the trustworthiness of that infrastructure.

It is yet unclear where the limits of mechanical discovery of insecure security protocol option selections is and if it is feasible to infer possible fixes (in the sense of slightly adapted option sets) from the discovery results. It is necessary to determine the limitations in order to know, how much manual specification is needed to augment the automated approach.

4 Conclusion

Pervasive computing leads to ever-more flexible security protocols with a multitude of options. Security validation needs to consider families of security protocols instead of only single security protocols, which leads to a huge increase of possible protocol instances that need to be validated. In this paper we therefore propose to take the options selected during run-time by the end-points into consideration, thereby focusing only on the protocol instances relevant in the particular situation. This novel concept opens a number of research questions: Is it feasible that each end-point validates the protocol properties it requires? How would the trustworthiness of the input to validation be assessed? As first steps in the direction of on-the-fly security protocol validation we need to specify what information is needed to describe a protocol instance and to determine the computational complexity of on-the-fly validation given the restriction to a choice of selected protocol options.

References

- [1] Alessandro Armando, David A. Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Paul Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, Sebastian Mödersheim, David von Oheimb, Michaël Rusinowitch, Judson Santiago, Mathieu Turuani, Luca Viganò, and Laurent Vigneron. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In Kousha Etessami and Sriram K. Rajamani, editors, *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285. Springer, 2005. http://dx.doi.org/10.1007/11513988_27.
- [2] Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, and Llanos Tobarra Abad. Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps. In *Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering (FMSE 2008)*. ACM Press, 2008.

- [3] Alessandro Armando and Luca Compagna. SAT-based Model-Checking for Security Protocols Analysis. *International Journal of Information Security*, 7(1):3–32, 2008.
- [4] B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. pages 82–96. *ieeecos*, 2001.
- [5] C.J.F. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, USA, Proc.*, volume 5123/2008 of *Lecture Notes in Computer Science*, pages 414–418. Springer, 2008.
- [6] D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.
- [7] R. M. Needham and M. D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. Technical Report CSL-78-4, Xerox Palo Alto Research Center, Palo Alto, CA, USA, 1978. Reprinted June 1982.
- [8] OASIS. Security Assertion Markup Language (SAML) v2.0. Available at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security, April 2005.