

Towards Validating Security Protocol Deployment in the Wild



Stelvio Cimato

Luca Compagna, Ulrich Flegel, Volkmar Lotz

SAPSE, Seattle (US), July 24th 2009

supported by the projects: **AVANTSSAR** (EU-funded) and THESEUS/TEXO (German-funded)

Agenda



- Motivation and context
- Problem statement and the new challenges
- Formalizing the problem

- Computing technology becoming increasingly pervasive and interconnected leads to shorter-lasting relationships between end-points with many different security requirements
- Rapid development of new service landscapes calls for standardized, yet highly flexible security protocols so to meet the multitude of requirements emerging from the different actors involved
- The more the protocol will be open and flexible, the more the actors will be able to interoperate by using it,
 - Typically, standards offer a multitude of options
- BUT the chances of unpredicted and undesired protocol deployments violating critical security desiderata increase significantly
- ▶ Need for on-the-fly verification of security protocol instances considering the actual choice of options and environment assumptions

- Formal analysis has been widely used in the area of security protocols
- Techniques are available to analyse large protocols (large number of protocol steps, messages and message fields)
 - E.g., AVISPA, ProVerif, and Scyther successfully applied to industrial relevant security protocol specifications like those proposed by the Internet Engineering Task Force (IETF)
- So far applied at design time on *protocol definitions*, i.e., reference scenarios where information on the specific selection of options and assumptions about the protocol execution and application context are missing

- Recent work [1] has shown how sensitive protocols are to their respective environment, even if the reference scenario is proven to be secure
- Example from [1]:
 - Google Apps offers a SAML-based Single-sign-on Identity Provider (IdP)
 - Absence of a few but critical fields in the authentication assertion generated by the IdP caused a security vulnerability
 - Vulnerability found through formal analysis

The chosen deployment options do have an impact on security

[1] A. Armando, R. Carbone, L. Compagna, J. Cuellar, L. Tobarra Abad. *Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps*. ACM FMSE 2008

- Chosen deployment options and context have to be seen as critical part of the security analysis and validation
- → shift security protocol verification from the design-time realm into deployment and even run-time
 - validation of protocol *deployment* instead of protocol *definition*
 - protocol *deployment*: actual instance of a protocol specification implemented by an entity with its specific selection of options and assumptions
- E.g., for Google it is more crucial to validate the SAML-based SSO solution as they deploy it, rather than the entire SSO standard
- Scaling traditional formal analysis to ensure on-the-fly 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

poses a certain number of challenges

Challenges (1)



- **Validation of protocol specification vs protocol definition:**
 - 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 and assumptions
- **Preferences, policies, constraints emerging by the actors:**
 - actors have preferences, policies, constraints that narrow down the freedom in the selection of options for the given execution environment
 - all this has to be considered during the on-the-fly formal analysis

Challenges (2)



- **Attacker threat model:**
 - the popular Dolev-Yao (DY) threat model cannot be systematically used here
 - the usual justification – protocol secure against DY certainly is secure against a less powerful, perhaps more realistic attacker – 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
 - E.g., if a certain communication channel is assumed unreadable for the attacker, an attacker unable to access that channel 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
- **On-the-fly validation (before executing the chosen security protocol instance):**
 - the protocol instance has to comply with the actor preferences still achieving its critical security desiderata.
 - where and how undertaking this on-the-fly 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

The classical validation problem



- Existing formal validation approaches able to establish if a protocol scenario S interleaved with the attacker threat model I achieves the protocol desiderata G
- Let P be a traditional security protocol supposed to achieve certain security desiderata G (e.g., secrecy of data, authentication, non-repudiation, etc)
- P can be defined as a set of role behaviors parametric on certain data parameters

$$\mathcal{P} = \{R_1(P_{1,1}, \dots, P_{1,k_1}), \dots, R_n(P_{n,1}, \dots, P_{n,k_n})\}$$

- E.g., the well-known Needham-Schroeder Public Key protocol can be defined in terms of the behavior of its roles initiator and responder where the role parameters range over actor identifiers, actor keys, actor databases, etc.
- Roles are played by actors e.g., alice, bob, etc.

The classical validation problem (cont.)



- An actor A playing role R of protocol P in a protocol session $\#$ and with parameters of R properly instantiated defines an *actor session*, in symbol $A(R\sigma_{A,\#}, P, \#)$
- An interleaving of actor sessions defines a *protocol scenario* S .
- E.g., a protocol scenario for NSPK could be

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

- Existing formal validation approaches able to establish if a protocol scenario S interleaved with the attacker threat model I achieves the protocol desiderata G

$$S, I \models G$$

- 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

$$\forall S. S, I \models G$$

The on-the-fly validation problem: Role Behavior and Actor Preferences



- Security protocols for modern environments are defined via protocol specifications and are employed by actors having their own preferences, policies and constraints
- A *protocol specification* PS defines a set of role specifications $\{R_1, \dots, R_n\}$ and offers a multitude of options whose configurations produce a family of likely similar, but not identical security protocols $\{P^1, \dots, P^m\}$
 - Example role specifications:
SAML 2.0 Service Provider (SP), SAML 2.0 Identity Provider (IdP)
 - Example security protocols:
 $P^1:(SP, IdP)$ with AuthRequest signed, $P^2:(SP, IdP)$ without AuthRequest not signed
- Each protocol P^i defines specific *role behaviors* for R_1, \dots, R_n . Let $rb(R_j, P^i)$ be the role behavior associated to the role specification R_j of protocol P^i
 - Example role behavior:
 $rb(SP, P^2)$ describes the SP not signing the AuthRequest
- Preferences, policies and constraints of an actor A are captured by the formula $ap(A)$
 - Example actor preferences:
actor A does not use digital signature

The on-the-fly validation problem: Protocol Scenario Specification



- Agent A that wants to play the role specification R in a session $\#$ can deploy any role behavior of R that satisfies $ap(A)$
 - Example:
 A with $ap(A)$ as above can deploy $rb(SP, P^2)$, but not $rb(SP, P^1)$
- Let $canplay(A, R, PS, ap(A))$ be the set of those protocols P^{l1}, \dots, P^{lk} such that $rb(R, P^{lj})$ complies with $ap(A)$ (for $j = 1, \dots, k$). Basically it says which protocols of PS agent A can play as R , without violating his/her preferences, policies, etc.
 - Example:
 $canplay(A, SP, SAMLSSO, ap(A))$ would comprise P^2 , but not P^1
- An *actor session specification* $A[R, PS, \#]$ denotes actor A that wants to play the role specification R in session $\#$
- A set of actor session specifications defines a *protocol scenario specification* SS

The on-the-fly validation problem (cont.)



- The on-the-fly validation problem:

Ensure compliance with actors' preferences, policies, etc

$$\forall A[R, PS, \#] \in SS. \forall P^i \in \bigcap_{A,R} canplay(A, R, PS, ap(A)).$$

Validation problem

$$\|\{A(rb(R, P^i)\sigma_{A,\#}, P^i, \#)\}, I \models G$$

Focus on a scenario specification

amounts to solving multiple standard validation problems, one for each suitable protocol deployment P^i complying with the respective actors' preferences and with the given scenario specification

- Any violation of the validation problem

$$\|\{A(rb(R, P^i)\sigma_{A,\#}, P^i, \#)\}, I \models G$$

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

- If $canplay(A, R, PS, ap(A))$ results in an empty set then the preferences expressed by the actors make them unable to deploy any of the protocols of PS

- Assessing the impact of a selected scenario and environment constraints on the security of a protocol specification
- Preferences, policies, constraints emerging by the actors
 - Languages are required that express these preferences and restrictions
 - While existing policy languages may be useful here, it remains to be investigated if they can express that information sufficient for computing, e.g., the *canplay* set
 - We expect that *canplay* may generate huge result sets, such that traditional validation techniques may run into problems
 - It is worth considering combining the validation techniques with policy reasoning techniques to narrow down to a manageable focus for validation
- Where and how performing the on-the-fly validation?
 - 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
 - For low-powered 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.
- 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)

Conclusions



- Pervasive computing leads to ever-more flexible security protocols with many options
- The chosen deployment options do have an impact on security!
- Formal validation to provide the required level of insurance
 - needs to consider families of security protocols instead of only single security protocols
 - huge increase of protocol instances to be validated
 - ▶ 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, e.g.,
 - Is it feasible that each endpoint validates the protocol properties it requires?
 - How would the trustworthiness of the input to validation be assessed?
- First steps in the direction of on-the-fly security protocol validation
 - specify what information is needed to describe a protocol instance and
 - determine the computational complexity of on-the-fly validation given the restriction to a choice of selected protocol options

Thank you!

