Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps

Alessandro Armando
AI-Lab, DIST – U. di Genova
Viale Causa 13
16145, Genova, Italy
armando@dist.unige.it

Roberto Carbone
AI-Lab, DIST – U. di Genova
Viale Causa 13
16145, Genova, Italy
carbone@dist.unige.it

Luca Compagna
SAP Research
805 Av. du Dr M. Donat
06250, Mougins, France
luca.compagna@sap.com

Jorge Cuellar
Siemens AG
Corporate Technology
D-80200 Munich, Germany
jorge.cuellar@siemens.com

Llanos Tobarra
I^{3}A, U. de Castilla-La Mancha
Campus Universitario s/n
02071, Albacete, Spain
mtobarra@dsi.uclm.es

ABSTRACT

Single-Sign-On (SSO) protocols enable companies to establish a federated environment in which clients sign in the system once and yet are able to access to services offered by different companies. The OASIS Security Assertion Markup Language (SAML) 2.0 Web Browser SSO Profile is the emerging standard in this context. In this paper we provide formal models of the protocol corresponding to one of the most applied use case scenario (the SP-Initiated SSO with Redirect/POST Bindings) and of a variant of the protocol implemented by Google and currently in use by Google's customers (the SAML-based SSO for Google Applications). We have mechanically analysed these formal models with SATMC, a state-of-the-art model checker for security protocols. SATMC has revealed a severe security flaw in the protocol used by Google that allows a dishonest service provider to impersonate a user at another service provider. We have also reproduced this attack in an actual deployment of the SAML-based SSO for Google Applications. This security flaw of the SAML-based SSO for Google Applications was previously unknown.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Protocol verification*; C.2.6 [**Computer-Communication Networks**]: Internetworking—*Standards, SAML, TLS*; D.2.4 [**Software Engineering**]: Software/Program Verification—*Formal methods, Model checking*; E.4 [**Coding and Information Theory**]: Formal models of communication; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages—*Process models, Program Analysis*; K.4.4 [**Computers and Society**]: Electronic Commerce—*Security*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication*

General Terms

Security, Verification

Keywords

Security Protocols, SAML Single Sign-On, Bounded Model Checking, SAT-based Model Checking

FMSE'08, October 27, 2008, Alexandria, Virginia, USA.

1. INTRODUCTION

The management of multiple user-names and passwords is not only an annoying aspect of the current Internet, it is also one of the most serious security weakness. Each system requires that a client registers himself in order to access to the services. But rather often a user is registered in several web sites under the same user-name and with the same or closely related passwords, which is not a secure practice. Or they often forgets their user-name and password and the user management system sends an unencrypted e-mail with these confidential data.

Single Sign-On (SSO) protocols tackle the problem by enabling companies to establish a federated environment in which clients sign in the environment once and yet are able to access to services offered by different companies. The *Security Assertion Markup Language* (SAML) 2.0 Web Browser SSO Profile (SAML SSO, for short) [15] is the emerging standard in this context: it defines an XML-based format for encoding security assertions as well as a number of protocols and bindings that prescribe how assertions should be exchanged in a variety of applications and/or deployment scenarios. SAML SSO is at the core of several SSO solutions like the Liberty Alliance project [13] and the Shibboleth Project [11]. In addition, established software companies base their SSO implementations on SAML SSO. This is the case of Google that developed a SAML-based Single Sign-On for its Google Apps Premier Edition [7], a service for using custom domain names with several Google

web applications (e.g. Gmail, Google Calendar, Talk, Docs and Sites).

The security of a SAML SSO solution critically depends on several assumptions (e.g. the trust relationships among the involved parties) and security mechanisms (e.g. the secure transport protocols used to exchange messages). The many security recommendations that are available throughout the bulky SAML specifications are useful in avoiding the most common security pitfalls but are of little help in ensuring their absence. It is therefore very difficult to achieve the needed level of assurance even for very simple instances of the protocol.

In this paper, we provide a formal model of the protocol corresponding to one of the most employed use case scenario of the SAML Web Browser SSO Profile: the SP-Initiated SSO with Redirect/POST Bindings. We have mechanically analysed it with SATMC, a state-of-the-art model checker for security protocols. In doing this we have extended previous work on SAML SSO (e.g. [8, 10]) *(i)* by considering the latest version of the SAML specifications (i.e. SAML 2.0), *(ii)* by relaxing the assumptions on the trustworthiness of the principals, and *(iii)* by using a more general model of the transport protocols.

We have also built a formal model of the protocol implemented in the SAML-based SSO for Google Apps and analysed it with SATMC. SATMC has revealed a severe security flaw in Google's variant of the protocol that allows a dishonest service provider to impersonate a user at another service provider. We have also reproduced this attack in an actual deployment of the SAML-based SSO for Google Applications. This security flaw of the SAML-based SSO for Google Applications was previously unknown. We have promptly reported this vulnerability to Google as well as to the Computer Emergency Response Team (CERT). At the time of this writing Google is instructing their customers to implement measures to mitigate potential exploits and has started offering a new version of the SSO service which does not suffer from the problem. CERT will soon release a vulnerability report describing the problem.

Structure of the paper.

In the next section we give a brief introduction of the SAML SSO. In Section 3 we describe the specification formalism we use to specify security protocols, the properties of the transport protocols and the security properties that the protocols are expected to meet. In Section 4 we present the results of our analysis of the SAML SSO with SATMC. In Section 5 we discuss some of the related work and in Section 6 we conclude.

2. THE SAML WEB BROWSER SSO PROTOCOLS

SAML specifications are based on the notions of assertions, protocols, bindings and profiles: A SAML *assertion* is an XML expression encoding a statement about a principal (also called subject). In this paper we will consider only a special type of assertions, called authentication assertions. An *authentication assertion* states that a given subject was authenticated by a particular means at a particular time. *Protocols* prescribe how assertions should be exchanged and *bindings* detail how assertions can be mapped into transport protocols, e.g. SOAP or HTTP. Finally, *profiles* define the use of SAML assertions, protocols and bindings so to meet some specific use case requirements (e.g. SSO). This is done to the minimum extent needed to guarantee the interoperability among different implementations. As a consequence, SAML SSO is a framework which needs to be instantiated according to the requirements posed by the application scenario and the available security mechanisms.

Three roles take part in the protocol: a client (C), an identity provider (IdP) and a service provider (SP). C, typically a web browser guided by a user, aims at getting access to a service or a resource provided by SP. IdP authenticates C and issues corresponding authentication assertions. Finally, SP uses the assertions generated by IdP to give C access to the requested service. A session is a particular run of the protocol, in which each role is played by a specific principal.

The Web Browser SAML 2.0 SSO profile [14] provides a wide variety of options, primarily having to do with two dimensions of choice: first whether the message flows are IdP-initiated or SP-initiated, and second, which bindings are used to deliver messages between the IdP and the SP. In this paper we report on the SP-initiated SSO using a Redirect Binding for the SP-to-IdP `<AuthnRequest>` message and a POST Binding for the IdP-to-SP `<Response>`. The protocol is depicted in Figure 1. In step S1, C asks SP to provide the resource located at the address URI. SP then initiates the *SAML Authentication Protocol* by sending C a redirect response directed to IdP containing an authentication request of the form AuthReq(ID,SP)—where ID is a string uniquely identifying the request—and the address URI of the resource. IdP then challenges C to provide valid credentials and—if the authentication succeeds—IdP builds an authentication assertion $AA = \text{AuthAssert}(\text{ID}, \text{C}, \text{IdP}, \text{SP})$ and places it into a response message $Resp = \text{Response}(\text{ID}, \text{SP}, \text{IdP}, \{AA\}_{\text{K}_{\text{IdP}}^{-1}})$, where $\{AA\}_{\text{K}_{\text{IdP}}^{-1}}$ is the assertion digitally signed with $\text{K}_{\text{IdP}}^{-1}$, the private key of IdP. (As unnecessary for our findings, we will not model how the authentication between C and IdP is performed, but we simply assume it is successful.) IdP then places *Resp* into an HTML form as a hidden form control and sends it back to C. For ease of use purposes, the HTML form is typically accompanied by script code that automatically posts the form to SP. This completes the protocol and SP can deliver the requested resource to C.

The protocol discussed above results from a considerable effort we put into a careful scrutiny and interpretation of the modular and open, but informal and bulky SAML 2.0 specifications.

Our interpretation of the standard seems to differ from the one of Google whose SAML-based Single Sign-On for Google Applications deviates from the above protocol for a few, seemingly minor simplifications in the messages exchanged:

(G1) ID and SP are not included in the authentication assertion, i.e. AA is $\text{AuthAssert}(\text{C}, \text{IdP})$ instead of $\text{AuthAssert}(\text{ID}, \text{C}, \text{IdP}, \text{SP})$;

(G2) ID, SP and IdP are not included in the response, i.e. *Resp* is $\text{Response}(\{AA\}_{\text{K}_{\text{IdP}}^{-1}})$ instead of $\text{Response}(\text{ID}, \text{SP}, \text{IdP}, \{AA\}_{\text{K}_{\text{IdP}}^{-1}})$.

The online static demo of the SAML-based SSO for Google Apps, available on Google at `http://code.google.com/apis/apps/sso/saml_static_demo/saml_demo.html`, illustrates each

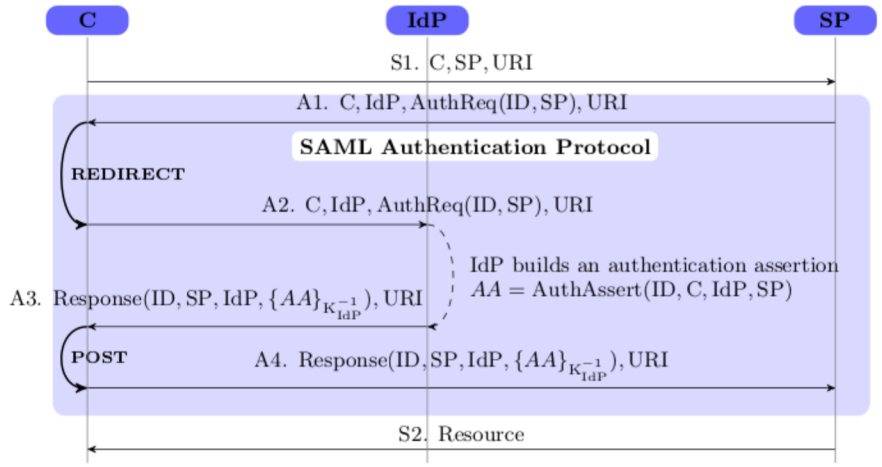

Figure 1: SP-Initiated SSO with Redirect/POST Bindings

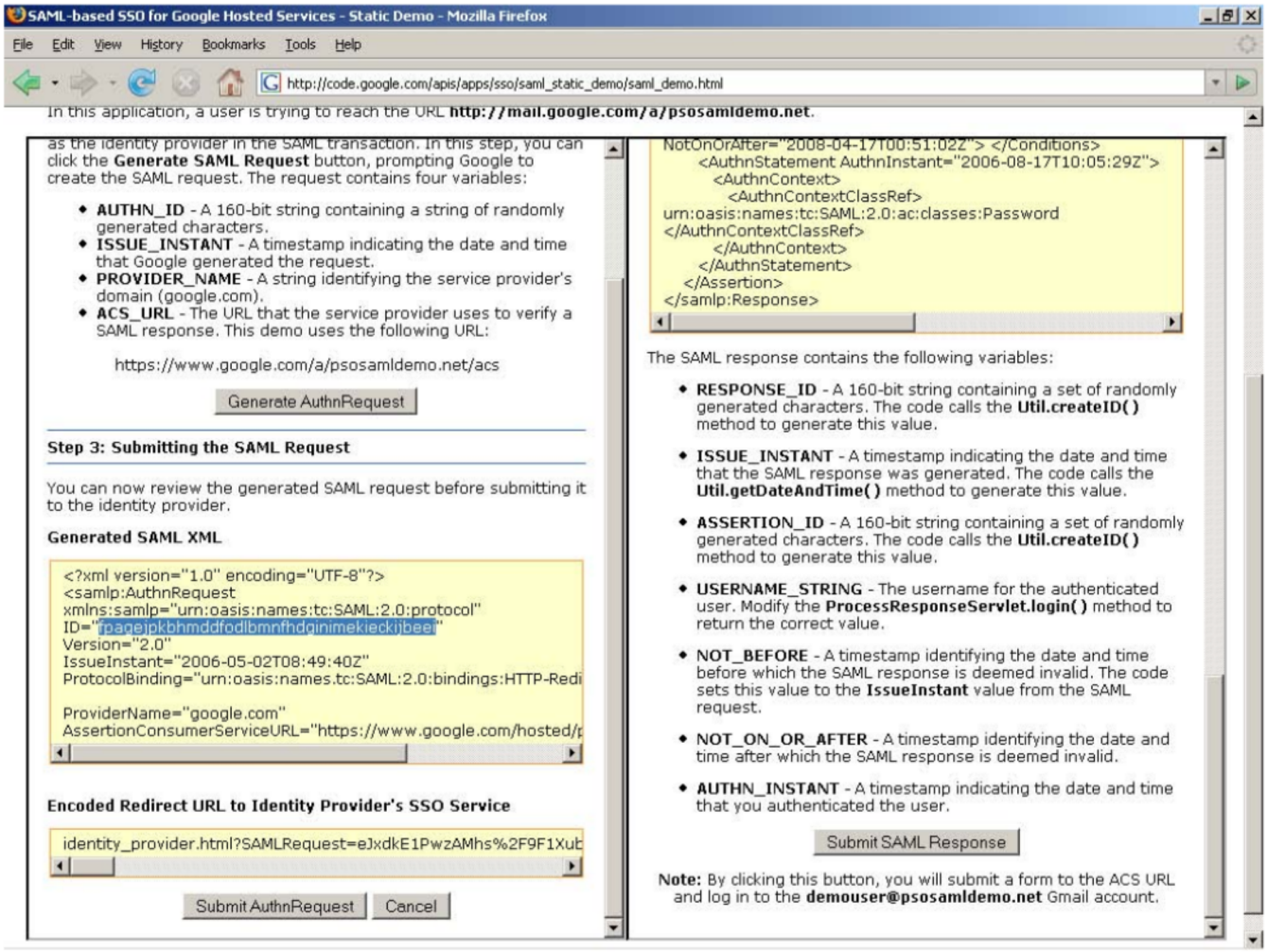

Figure 2: Screenshot of the online static demo of the SAML-based SSO for Google Apps

step of the SAML workflow between Google and a partner IdP and allowed us to spot these divergences. Figure 2 depicts a screenshot of this static demo. The frame on the left hand side represents the SP while the one on the right hand side captures the IdP. The authentication response generated by the IdP in the demo (see Figure 4) contains neither the ID nor the SP fields occurring in the authentication request created by the SP (see Figure 3). As we will see in

Section 4 these changes compromise the security of the protocol.[1]

The security of the SAML SSO Protocol relies on a number of assumptions about the trustworthiness of the principals involved as well as the security of the transport protocols employed that we summarise hereafter.

Trust Assumptions. The protocol assumes that IdP is trustworthy for both C and SP, but SP is not assumed to be trustworthy. In accordance with this in our analysis we will consider protocol sessions in which C and SP can be played by the intruder, while we assume that IdP is played by an honest agent.

Assumptions on the communication channels. Communications between the parties are subject to the following assumptions:

(A1) the communication between C and SP is carried over a unilateral SSL 3.0 or TLS 1.0 channel (henceforth SSL/TLS), established through the exchange of a valid certificate (from SP to C); and

(A2) the communication between C and IdP is carried over a unilateral SSL/TLS channel that becomes bilateral once C authenticates itself on IdP. This is established through the exchange of a valid certificate (from IdP to C) and of valid credentials (from C to IdP).

Under the above assumptions the protocol is expected to meet the following security properties: *(i)* SP authenticates C, i.e., at the end of its protocol run SP believes it has been talking with C; and *(ii)* Resource must be kept secret between C and SP.

3. FORMALISATION OF THE SAML SSO PROTOCOL

We focus on the problem of determining whether the concurrent execution of a finite number of sessions of the protocol enjoys the expected security properties in spite of the interference of a malicious intruder. We show that this problem can be recast into a model checking problem of the form

$$M \models (C_I \Rightarrow G) \tag{1}$$

where M is a labelled transition system modelling the initial state of the system and the behaviours of the principals (including the intruder), C_I is a conjunction of LTL formulae (henceforth called *LTL constraints*) constraining the allowed behaviours of the intruder, and G is an LTL formula stating the security property that the protocol is expected to enjoy.

A *fact* is a ground (i.e. variable free) atomic formula of a first-order language with sorts. If T is a sort, then *the domain of* T is a set of ground (i.e. variable-free) terms of sort T. Since our model checking technique carries out a bounded analysis of the protocols, we assume that the domains of all sorts are finite. (The computation of finite over-approximation of the domains can be done in polynomial time by carrying out a static analysis of the protocol [3].)

[1]At the time of this writing Google has been promptly informed by us about our findings and it has already released a new version of the SAML-based SSO for Google Apps not suffering from the attack reported by us. The online static demo hosted at Google is thus likely to be changed accordingly.

We represent the states of M as sets of *facts* and its transitions as *set-rewriting rules* that define mappings between sets of facts. If S is a set of facts, then we interpret the facts in S as the propositions holding in the state represented by S, all other facts being false in that state (closed-world assumption). Let $(L \xrightarrow{\rho} R)$ be (an instance of) a rewrite rule and S be a set of facts. If $L \subseteq S$ then we say that ρ is applicable in S and that $S' = \mathrm{app}_\rho(S) = (S \setminus L) \cup R$ is the state resulting from the execution of ρ in S. A *path* π is an alternating sequence of states and rules $S_0\rho_1S_1 \ldots S_{n-1}\rho_nS_n$ such that $S_i = \mathrm{app}_{\rho_i}(S_{i-1})$ (i.e. S_i is a state resulting from the execution of ρ_i in S_{i-1}), for $i = 1, \ldots, n$. If, additionally, $S_0 \subseteq \mathcal{I}$, then we say that the path is *initialised*. Let $\pi = S_0\rho_1S_1 \ldots S_{n-1}\rho_nS_n$ be a path. We define $\pi(i) = S_i$ and $\pi_i = S_i\rho_{i+1}S_{i+1} \ldots S_{n-1}\rho_nS_n$. $\pi(i)$ and π_i are the i-th state of the path and the suffix of the path starting with the i-th state respectively. We also assume that paths have infinite length. This can be always obtained by adding stuttering transitions to the transition system.

The language of LTL we consider uses facts and equalities over ground terms as atomic propositions, the usual propositional connectives (namely, $\neg$, $\vee$) and the temporal operators **F** (eventually) and **O** (once). Let π be an initialised path of M, an LTL formula f *is valid on* π, written $\pi \models f$, if and only if $(\pi, 0) \models f$, where $(\pi, i) \models f$ (f holds in π at time i) is inductively defined as follows:

$$\begin{array}{ll}
(\pi,i) \models f & \text{iff } f \in \pi(i) \\
(\pi,i) \models (t_1 = t_2) & \text{iff } t_1 \text{ and } t_2 \text{ are the same term} \\
(\pi,i) \models \neg f & \text{iff } (\pi,i) \not\models f \\
(\pi,i) \models (f \vee g) & \text{iff } (\pi,i) \models f \text{ or } (\pi,i) \models g \\
(\pi,i) \models \mathbf{F}f & \text{iff exists } j \in [i,\infty).(\pi,j) \models f \\
(\pi,i) \models \mathbf{O}f & \text{iff } \exists j \in [0,i].(\pi,j) \models f
\end{array}$$

In the sequel we use $(f \wedge g)$, $(f \Rightarrow g)$ and $\mathbf{G}f$ as abbreviations of $\neg(\neg f \vee \neg g)$, $(\neg f \vee g)$ and $\neg\mathbf{F}\neg f$ respectively. Moreover, if X is a variable of sort T, then with $\mathrm{dom}(X)$ we denote the domain of T and use $\forall X.f$ and $\exists X.f$ as an abbreviation of the formulae $\bigwedge_{t \in dom(X)} f[t/X]$ and $\bigvee_{t \in dom(X)} f[t/X]$ respectively, where $f[t/X]$ is the formula obtained by substituting all free occurrences of X in f with the term t. Finally we use $\forall(f)$ and $\exists(f)$ as abbreviations of $\forall X_1 \ldots \forall X_n.f$ and $\exists X_1 \ldots \exists X_n.f$ respectively, where $X_1, \ldots, X_n$ are the free variables of the formula f.

In Section 3.1 we show how the behaviours of the principals can be specified using a set-rewriting formalism. This amounts to specifying the model M of (1). In Section 3.2 we show how assumptions on the behaviour of the intruder can be specified by means of LTL constraints (corresponding to the C_I constraints of (1)). Finally in Section 3.3 we show how the security properties that the protocol is expected to enjoy can be specified by means of LTL formulae (corresponding to the formula G in (1)).

3.1 Formalising the Behaviour of Principals

Facts are expressions of the form given in the left column of Table 1 and whose informal meaning is described in the right column. The constant `i` of sort AGENT is used to denote the intruder. If S is a set of facts representing a state, then the state of the honest principal a is represented by all the facts of the form $\mathtt{state}_R(j, a, es, s)$ (called *state-facts*) occurring in S. By construction, for each session s and for each principal a there exists at most one fact of the form $\mathtt{state}_R(j, a, es, s)$ in S. Notice that this does not prevent an

```
<samlp:AuthnRequest ID="fpagejpkbhmddfodlbmnfhdginimekieckijbeei" Version="2.0" IssueInstant="2006-05-02T08:49:40Z"
ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" ProviderName="google.com"
AssertionConsumerServiceURL="https://www.google.com/hosted/psosamldemo.net/acs"/>
```

Figure 3: SAML Authentication Request from the online static demo

```
- <samlp:Response ID="hedangifkfodeigidaeijpdnfjkfbnegddealebo" IssueInstant="2006-08-17T10:05:29Z" Version="2.0">
  - <Signature>
    + <SignedInfo></SignedInfo>
    + <SignatureValue></SignatureValue>
    + <KeyInfo></KeyInfo>
    </Signature>
  - <samlp:Status>
      <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
    </samlp:Status>
  - <Assertion ID="dojnoaponicbieffopfdecilinaepodfimmkpjij" IssueInstant="2003-04-17T00:46:02Z" Version="2.0">
      <Issuer>https://www.opensaml.org/IDP </Issuer>
    - <Subject>
        <NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"> demouser </NameID>
        <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
      </Subject>
      <Conditions NotBefore="2003-04-17T00:46:02Z" NotOnOrAfter="2008-04-17T00:51:02Z"> </Conditions>
    - <AuthnStatement AuthnInstant="2006-08-17T10:05:29Z">
      - <AuthnContext>
          <AuthnContextClassRef> urn:oasis:names:tc:SAML:2.0:ac:classes:Password </AuthnContextClassRef>
        </AuthnContext>
      </AuthnStatement>
    </Assertion>
  </samlp:Response>
```

Figure 4: SAML Authentication Response from the online static demo

agent from playing different roles in different sessions. No state-fact is associated with the intruder.

While the initial state of the system defines the initial knowledge of the intruder (usually including its cryptographic material, various agent identifiers, their public keys and the communication channels on which the intruder has some control) and the initial state of all the honest principals involved in the protocol sessions considered, rewriting rules specify the evolution of the system. The behaviour of honest participants is specified by rules of the form:

$$\mathtt{sent}(\mathtt{RS},\mathtt{B},\mathtt{A},\mathtt{M},\mathtt{Ch}) \xrightarrow{\mathtt{receive}(\mathtt{A},\mathtt{B},\mathtt{RS},\mathtt{M},\mathtt{Ch})} \mathtt{rcvd}(\mathtt{A},\mathtt{B},\mathtt{M},\mathtt{Ch}) \quad (2)$$

$$\mathtt{rcvd}(\mathtt{A},\mathtt{B},\mathtt{M},\mathtt{Ch}).\mathtt{state}_r(j,\mathtt{A},es,\mathtt{S}) \xrightarrow{\mathtt{send}_j(\mathtt{A},\mathtt{B},\mathtt{B1},\ldots,\mathtt{S})} \mathtt{sent}(\mathtt{A},\mathtt{A},\mathtt{B1},\mathtt{M1},\mathtt{Ch1}).\mathtt{state}_r(l,\mathtt{A},es',\mathtt{S}) \quad (3)$$

Rule (2) models the reception of a message by an honest agent, whereas rule (3) models the processing of a previously received message and the sending of the next protocol message. Notice that rule (3) may take slightly different forms depending on the type of the protocol step modelled. For instance, if $j = 1$ and $\mathtt{A}$ plays the role of initiator (i.e. the principal that sends the first message of the protocol), the fact $\mathtt{rcvd}(\mathtt{A},\mathtt{B},\mathtt{M},\mathtt{Ch})$ is not included in the left hand side of the rule. Similarly, if a fresh term must be generated and sent, then $\mathtt{c}(\mathtt{N})$ and $\mathtt{c}(\mathtt{s}(\mathtt{N}))$ are included in the left and right hand sides of the rule respectively. A further variant is necessary when the step involves either a membership test or an update of a set of elements. In this case the facts of the form $\mathtt{contains}(db, m)$ must be properly included in the rules.

To illustrate we consider the protocol steps in which the authentication request sent by SP and received by C is redirected to IdP. This transition is modelled by rule (2) and by the following one, in which C forwards to IdP the message A1 received from SP (see Figure 1):

$$\mathtt{rcvd}(\mathtt{C},\mathtt{SP},\langle \mathtt{C},\mathtt{IdP},\mathtt{authReq}(\mathtt{ID},\mathtt{SP}),\mathtt{URI}\rangle,\mathtt{SP2C}).\mathtt{state}_\mathtt{C}(2,\mathtt{C},[\mathtt{SP},\mathtt{URI},\ldots],\mathtt{S}) \xrightarrow{\mathtt{send}_2(\mathtt{C},\ldots,\mathtt{S})} \mathtt{sent}(\mathtt{C},\mathtt{C},\mathtt{IdP},\langle \mathtt{C},\mathtt{IdP},\mathtt{authReq}(\mathtt{ID},\mathtt{SP}),\mathtt{URI}\rangle,\mathtt{C2SP}).\mathtt{state}_\mathtt{C}(3,\mathtt{C},[\mathtt{IdP},\mathtt{ID},\mathtt{SP},\mathtt{URI},\ldots],\mathtt{S})$$

where $\mathtt{SP2C}$ and $\mathtt{C2SP}$ are the channels used by $\mathtt{C}$ and $\mathtt{SP}$ to communicate with each other. (Channels in our model are unidirectional: if x and y are principals, then $x2y$ denotes a

Table 1: Facts and their informal meaning

Fact	Meaning
$\mathtt{state}_r(j, a, es, s)$	Principal a, playing role r, is ready to execute step j in session s of the protocol, and es is a list of expressions representing the internal state of a and thus affecting her future behaviour.
$\mathtt{ik}(m)$	The intruder knows message m.
$\mathtt{sent}(rs, b, a, m, ch)$	Principal rs has sent message m on channel ch to principal a pretending to be principal b.
$\mathtt{rcvd}(a, b, m, ch)$	Message m (supposedly sent by principal b) has been received on channel ch by principal a, but a has not processed it yet.
$\mathtt{c}(n)$	Term n is the current value of the counter used to construct fresh terms. This value is incremented every time a fresh term is used.
$\mathtt{contains}(db, m)$	Message m is contained into set db. Sets are used, e.g., to share data between honest principals.

channel used by x to send messages to y and $y2x$ denotes a channel used by y to send messages to x.)

The abilities of the intruder are modelled according to the standard Dolev-Yao attacker [6] by the following rules:

$$\mathtt{ik(M).ik(A).ik(B).ik(Ch)} \xrightarrow{\mathtt{fake(A,B,M,Ch)}} \mathtt{sent(i,A,B,M,Ch).ik(M).ik(A).ik(B).ik(Ch)}$$

$$\mathtt{sent(A,A,B,M,Ch).ik(Ch)} \xrightarrow{\mathtt{intercept(A,B,M,Ch)}} \mathtt{rcvd(i,A,M,Ch).ik(M).ik(Ch)}$$

$$\mathtt{sent(A,A,B,M,Ch).ik(Ch)} \xrightarrow{\mathtt{overhear(A,B,M,Ch)}} \mathtt{sent(A,A,B,M,Ch).rcvd(i,A,M,Ch).ik(M).ik(Ch)}$$

For instance, the first rule models the intruder that exploits its knowledge to impersonate an agent A in sending a message M to an agent B on a communication channel Ch. Finally, the inferential capabilities of the intruder are modelled by the following rules (where K is either K or $\mathtt{K}^{-1}$, and $\overline{K}$ is K if $K = \mathtt{K}^{-1}$ and $\overline{K}$ is $\mathtt{K}^{-1}$ otherwise):

$$\mathtt{ik(M).ik}(K) \xrightarrow{\mathtt{encrypt}(K,\mathtt{M})} \mathtt{ik(M).ik}(K)\mathtt{.ik}(\{\mathtt{M}\}_K)$$

$$\mathtt{ik}(\{\mathtt{M}\}_K)\mathtt{.ik}(\overline{K}) \xrightarrow{\mathtt{decrypt}(K,\mathtt{M})} \mathtt{ik}(\{\mathtt{M}\}_K)\mathtt{.ik}(\overline{K})\mathtt{.ik(M)}$$

$$\mathtt{ik}(\mathtt{M}_1)\mathtt{.ik}(\mathtt{M}_2) \xrightarrow{\mathtt{pairing}(\mathtt{M}_1,\mathtt{M}_2)} \mathtt{ik}(\mathtt{M}_1)\mathtt{.ik}(\mathtt{M}_2)\mathtt{.ik}(\langle \mathtt{M}_1, \mathtt{M}_2\rangle)$$

$$\mathtt{ik}(\langle \mathtt{M}_1, \mathtt{M}_2\rangle) \xrightarrow{\mathtt{decompose}(\mathtt{M}_1,\mathtt{M}_2)} \mathtt{ik}(\langle \mathtt{M}_1, \mathtt{M}_2\rangle)\mathtt{.ik}(\mathtt{M}_1)\mathtt{.ik}(\mathtt{M}_2)$$

3.2 Formalising the Communication Channels

As the security of SAML SSO ultimately depends on the security of the transport protocols used to exchange the messages, special care must be paid in modelling the communication channels. We do this by constraining the behaviour of the intruder through a number of LTL formulae each modelling a specific security property that the underlying communication channel is expected to enjoy.

Confidential channels.

A channel ch provides confidentiality if its output is exclusively accessible to a given receiver p. In our model this amounts to requiring that in every state S if a fact $\mathtt{rcvd}(a, b, m, ch) \in S$, then a has exclusive access to the channel. Thus, the condition that *channel ch is confidential to principal p* can be formalised by the following formula:

$$\mathit{confidential}(ch, p) := \mathbf{G}\,\forall(\mathtt{rcvd}(A, B, M, ch) \Rightarrow A = p)$$

Authentic channels.

A channel provides authenticity if its input is exclusively accessible to a specified sender. The condition that *channel ch is authentic for principal p* can be formalised by the following constraint:

$$\mathit{authentic}(ch, p) := \mathbf{G}\,\forall(\mathtt{sent}(RS, A, B, M, ch) \Rightarrow (A = p \wedge RS = p))$$

A run of SSL/TLS in which one of the two principals does not have a valid certificate yields a unilateral authentic and confidential channel. In order to model this type of channel we relax the notions of confidential and authentic channels as follows.

Weakly confidential channels.

A channel provides weak confidentiality if its output is exclusively accessible to a single, yet unknown, receiver. In our model this amounts to requiring that, for every state S, if a fact $\mathtt{rcvd}(a, b, m, ch) \in S$, then in all the successor states the rcvd facts with channel ch must have a as recipient. The condition that *channel ch is weakly confidential* can then be formalised by the following formula:

$$\mathit{weakly\_confidential}(ch) := \mathbf{G}\,\forall((\mathtt{rcvd}(A, B, M, ch) \wedge \mathbf{F}\,\mathtt{rcvd}(A', B', M', ch)) \Rightarrow A = A')$$

Weakly authentic channels.

A channel provides weak authenticity if its input is exclusively accessible to a single, yet unknown, sender. In our model this amounts to requiring that, for every state S, if a fact $\mathtt{sent}(rs, a, b, m, ch) \in S$, then in all the succeeding states the sent facts with channel ch must have rs as real sender and a as official sender. The condition that *channel ch is weakly authentic* can then be formalised by the following constraint:

$$\mathit{weakly\_authentic}(ch) := \mathbf{G}\,\forall((\mathtt{sent}(RS, A, B, M, ch) \wedge \mathbf{F}\,\mathtt{sent}(RS', A', B', M', ch)) \Rightarrow (A = A' \wedge RS = RS'))$$

A run of SSL/TLS in which principal y has a valid certificate but principal x does not, is then modelled by a pair of channels $x2y$ and $y2x$, where $x2y$ is confidential to y and weakly authentic and $y2x$ is weakly confidential and authentic for y with the additional requirement that the principal sending messages on $x2y$ is the same principal that receives messages from $y2x$. This can be formalised by adding the

following constraints to C_I:

$$unilateral\_confidential\_authentic(x, y, x2y, y2x) := (confidential(x2y, y) \wedge weakly\_authentic(x2y) \wedge weakly\_confidential(y2x) \wedge authentic(y2x, y) \wedge \mathbf{G}\,\forall(\mathbf{F}\,\mathtt{sent}(RS, x, y, M, x2y) \wedge \mathbf{F}\,\mathtt{rcvd}(R, y, M', y2x)) \Rightarrow RS = R))$$

To illustrate the usage of the above constraints let us consider the SAML SSO and its security recommendations in matter of communication channels (cf. Section 2). Assumption (A1) requires that the message exchanges between C and SP are carried over unilateral SSL/TLS channels. Assumption (A2) imposes that the message from C to IdP is sent over a confidential channel, while the message from IdP to C is sent over a confidential and authentic channel. For each session s, this amounts to including the following constraints in C_I:

$$unilateral\_confidential\_authentic(c, sp, c2sp_{(s)}, sp2c_{(s)}), \tag{4}$$

$$confidential(c2idp_{(s)}, idp),\ authentic(idp2c_{(s)}, idp),\ confidential(idp2c_{(s)}, c) \tag{5}$$

where c, sp, and idp are the agents playing in session s the roles C, SP, and IdP respectively. (Clearly, constraints 4 and 5 capture assumptions (A1) and (A2) respectively.)

3.3 Security Properties

Authentication.

We base our definition of authentication on Lowe's notion of *non-injective agreement* [12]: whenever principal b (playing role R_2) completes a run of the protocol apparently with principal a (playing role R_1), then *(i)* a has previously been running the protocol apparently with b, and *(ii)* the two agents agree on m. We then say that b *authenticates* a *on* m *in session* s if and only if the following formula holds:

$$authentication(b, a, m, s) := \mathbf{G}\,\forall(\mathtt{state}_{r_2}(j_2, b, [a, \ldots, m, \ldots], s) \Rightarrow \exists\,\mathbf{O}\,\mathtt{state}_{r_1}(j_1, a, [b, \ldots, m, \ldots], s)) \tag{6}$$

where j_1 is the protocol step in which m is sent by an agent playing role r_1 and j_2 is the last protocol step of an agent playing role r_2.

Secrecy.

The secrecy of a message m is guaranteed if, and only if, the intruder does not know m. This is formalised by the following formula:

$$secret(m) := \mathbf{G} \neg \mathtt{ik}(m)$$

To illustrate an application of the above formulae let us consider again the SAML SSO and the security properties it is expected to enjoy (cf. end of Section 2). For each protocol session s, this amounts to including the following formulae in G:

$$authentication(sp, c, uri, s)$$

$$secret(resource) \text{ provided that } \mathtt{i} \notin \{c, sp\}$$

where c, sp, and idp are the agents playing in session s the roles C, SP, and IdP respectively, while uri and $resource$ are the data values of URI and Resource as exchanged by the agents in s. (Clearly, the intruder is entitled to access $resource$ if, and only if, it is playing in s one of those agents that is legitimated to do so i.e., the intruder is either c or sp.)

4. MODEL CHECKING OF THE SAML SSO PROTOCOL

We have mechanically analysed the formal model of the SAML SSO Protocol presented in Section 3 using SATMC, a state-of-the-art model checker for security protocols. At the core of SATMC lies a procedure that automatically generates a propositional formula whose satisfying assignments (if any) correspond to counterexamples (i.e. execution traces of M that falsify $(C_I \Rightarrow G)$) of length bounded by some integer k. Finding attacks (of length k) on the protocol therefore boils down to solving propositional satisfiability problems. SATMC relies on state-of-the-art SAT solvers for this task which can handle propositional satisfiability problems with hundreds of thousands of variables and clauses or more. SATMC can be instructed to perform an iterative deepening on k. More details on SATMC can be found in [3, 2].

By running SATMC against the formal model of the protocol implemented in the SAML-based Single Sign-On for Google Applications, we have found the attack depicted in Figure 5. In the attack, `bob` initiates a session of the protocol to access a resource provided by the (malicious) service provider `i` that in parallel starts a new session of the protocol with `google` pretending to be `bob` (indicated as `i(bob)`) and that mischievously reuses the authentication assertion received by `bob` (cf. step A4) to trick `google` into believing he is `bob`. The attack completes with the delivery of the resource (whose access should be reserved to `bob`) to `i`. This attack represents a violation of the two security properties that the protocol is expected to enjoy (cf. Section 2 and Section 3.3).

Notice that two protocol sessions sharing the same IdP are sufficient for a malicious SP to mount this attack and gain access to a resource of another SP under the identity of an unaware user. It is easy to imagine realistic scenarios in which this flaw can be exploited by a dishonest SP. To illustrate consider the scenario in which the AI-Lab has registered to the SAML SSO of Google to make available to its members Google's applications (e.g. Gmail, Google Calendar) without any additional authentication burden. Basically, once authenticated into the AI-Lab server, any user can transparently access to Google's remote applications. (AI-Lab is serving as IdP in this scenario.) Now, any other SP offering the very same SAML SSO solution as Google and attractive enough to convince the AI-Lab to include one of its remote services (e.g. free access to online scientific books) is able to mount the above attack and thus to impersonate any user of the AI-Lab IdP that accesses its resources at any Google Application.

We have reproduced the above attack in an actual deployment of the SAML-based SSO for Google Applications. To this end, we registered the `ai-lab.it` domain at the SAML SSO service and provided Google with the public key of the IdP of the AI-Lab. We then implemented a Java Servlet simulating the behaviour of a dishonest SP called BadSP. Af-

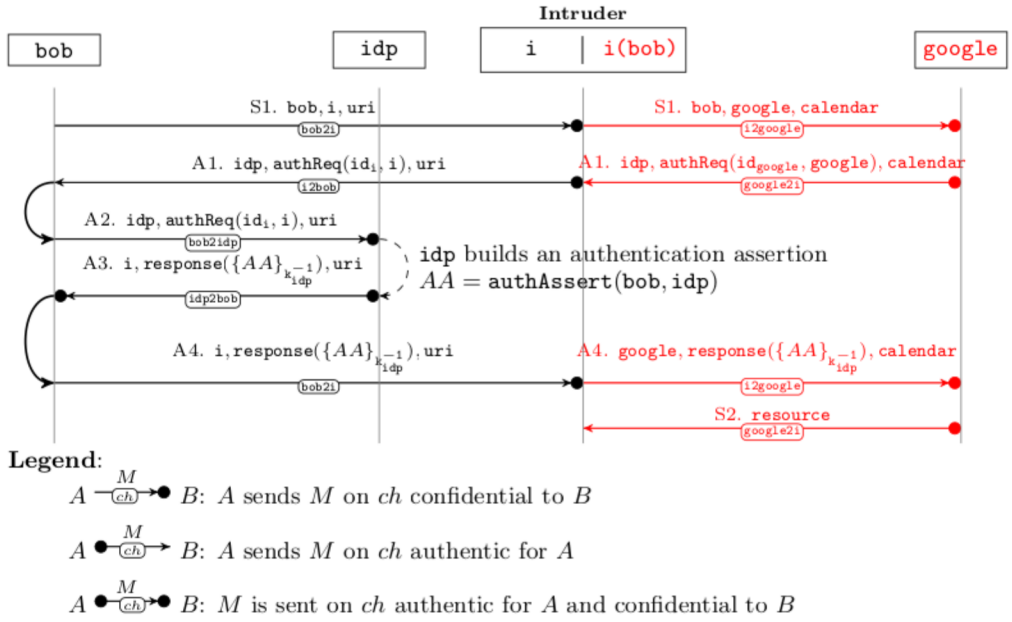

Figure 5: Attack on the SAML-based SSO for Google Applications

ter receiving the request for a resource from a web browser used by an AI-Lab's member, BadSP constructs a SAML Authentication Request and sends it back to the browser. BadSP then waits for the Response from the browser (this response is obtained from the IdP of the AI-Lab), parses the Response, and composes the fake Response for Google. The attack succeeds and Google logs BadSP into Google Applications as the AI-Lab member.

It is immediate to see that the attack originates from one of the simplifications that Google adopted in its SAML SSO solution and namely from simplification (G1) that deprives the authentication assertion of both the ID and SP fields (cf. Section 2). In fact, by performing a similar analysis on the standard SP-Initiated SSO with Redirect/POST Bindings, no attacks have been reported by SATMC despite the several protocol scenarios considered.

5. RELATED WORK

Pfitzmann et al [16, 17, 9] lay the theoretical basis for a rigorous analysis of web-based federated identity-management protocols (e.g., the Single-Sign-on protocol proposed by Liberty Alliance in 2002). They do not develop mechanised formal analyses, but they provide precise protocol descriptions, discussing their security vulnerabilities and possible preventive measures. Some of these results have been fed into the Liberty Alliance project and indirectly into the SAML 2.0 standard that according to our analysis does not suffer from those vulnerabilities. Still, producing attack-free SSO solutions does not seem an easy task as the Google example shows. More precise protocol descriptions (see, e.g., the BBAE protocol in [17]) might serve as deterrent and would make easier undertaking a formal analysis. In this respect, it is worth mentioning [4] where the idea is to couple real security protocol reference implementations with formal model specifications that can be then validated by means of state-of-the-art verifiers such as ProVerif [5] and AVISPA [1].

[8] and [10] investigate more specifically on SAML Single Sign-On. A security analysis of the SAML SSO Browser/Artifact Profile is presented in [8]. The paper (based on version 1.0 of the SAML specifications) shows that the treatment of the security aspects in the specifications may lead to flawed implementations and/or deployments of the protocol. Our work is based on version 2.0 of the SAML specifications which provides a more detailed treatment of the security aspects of the protocols. The work most closely related to ours is [10]. The authors provide a formal model of the SAML SSO Browser/Artifact Profile (SAML version 1.0) and describe the results of the analysis obtained by running a static analysis tool against several models of the protocol obtained by using communication channels of different strength. In our work we have focused on a single model of the protocol based on secure channels as they appear to be recommended by the latest version of the SAML specifications. In [10] it is assumed that the attacker cannot play the role of the SP. This assumption is not realistic as service providers do not necessarily trust each other. It is worth noticing that the adoption of this assumption would have prevented us from detecting the attack on the SAML-based SSO for Google Applications we reported in Section 4.

6. CONCLUSIONS

We have applied model-checking techniques to the analysis of SAML SP-Initiated SSO with Redirect/POST Bindings and to the protocol which is distributed and used by Google's partner companies to get full control over the authorization and authentication of hosted user accounts that can access web-based applications like Gmail or Google Calendar. Our analysis has not reported any attack on the model corresponding to (our interpretation of) the SAML 2.0 standard specifications despite the several protocol scenarios considered, but has unveiled a previously unknown attack in the Google's implementation. In line with respon-

sible disclosure principles, we have promptly reported this attack to Google and to the Computer Emergency Response Team (CERT). At the time of this writing Google is instructing their customers to implement measures to mitigate potential exploits of this vulnerability and has released a new version of the SSO service which does not suffer from the attack. CERT will soon release a vulnerability report describing the problem. This paper extends previous work by considering the latest version of the SAML specifications (i.e. SAML 2.0), by relaxing the assumptions on the trustworthiness of the principals and by using a more general model of the transport protocols.

7. ACKNOWLEDGMENTS

This work was partially supported by the FP7-ICT-2007-1 Project no. 216471, "AVANTSSAR: Automated Validation of Trust and Security of Service-oriented Architectures" (`www.avantssar.eu`).

The authors would like to thank the reviewers and our colleagues Luca Viganò and Volkmar Lotz for the valuable comments that helped to improve the paper.

8. REFERENCES

[1] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Heám, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In *Proceedings of the 17th International Conference on Computer Aided Verification (CAV'05)*. Springer-Verlag, 2005. Available at `www.avispa-project.org`.

[2] A. Armando, R. Carbone, and L. Compagna. LTL model checking for security protocols. In *20th IEEE Computer Security Foundations Symposium (CSF20)*, Venice (Italy), July 2007.

[3] A. Armando and L. Compagna. SAT-based Model-Checking for Security Protocols Analysis. *International Journal of Information Security*, 7(1):3–32, January 2008.

[4] K. Bhargavan, C. Fournet, A. D. Gordon, and N. Swamy. Verified implementations of the information card federated identity-management protocol. In *ASIACCS*, pages 123–135, 2008.

[5] B. Blanchet. From secrecy to authenticity in security protocols. In *In 9th International Static Analysis Symposium (SAS'02)*, pages 342–359. Springer, 2002.

[6] D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.

[7] Google. Web-based reference implementation of SAML-based SSO for Google Apps. `http://code.google.com/apis/apps/sso/saml_reference_implementation_web.html`, 2008.

[8] T. Groß. Security analysis of the SAML Single Sign-on Browser/Artifact profile. In *Proc. 19th Annual Computer Security Applications Conference*. IEEE, Dec. 2003.

[9] T. Groß, B. Pfitzmann, and A.-R. Sadeghi. Browser model for security analysis of browser-based protocols. In S. D. C. di Vimercati, P. F. Syverson, and D. Gollmann, editors, *ESORICS*, volume 3679 of *Lecture Notes in Computer Science*, pages 489–508. Springer, 2005.

[10] S. M. Hansen, J. Skriver, and H. R. Nielson. Using static analysis to validate the SAML single sign-on protocol. In *WITS '05: Proceedings of the 2005 workshop on Issues in the theory of security*, pages 27–40, New York, NY, USA, 2005. ACM Press.

[11] Internet2. Shibboleth Project. Available at `http://shibboleth.internet2.edu/`, 2007.

[12] G. Lowe. A hierarchy of authentication specifications. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop (CSFW'97)*, pages 31–43. IEEE Computer Society Press, 1997.

[13] OASIS. Identity Federation. Liberty Alliance Project. Available at `http://www.projectliberty.org/resources/specifications.php`, 2004.

[14] OASIS. Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0. Available at `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security`, March 2005.

[15] OASIS. Security Assertion Markup Language (SAML) v2.0. Available at `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security`, April 2005.

[16] B. Pfitzmann and M. Waidner. Analysis of Liberty Single-Sign-on with Enabled Clients. *IEEE Internet Computing*, 7(6):38–44, 2003.

[17] B. Pfitzmann and M. Waidner. Federated identity-management protocols. In B. Christianson, B. Crispo, J. A. Malcolm, and M. Roe, editors, *Security Protocols Workshop*, volume 3364 of *Lecture Notes in Computer Science*, pages 153–174. Springer, 2003.