LTL Model Checking for Security Protocol Analysis

Alessandro Armando

Artificial Intelligence Laboratory (AI-Lab)
Dipartimento di Informatica Sistemistica e Telematica (DIST)
University of Genova, Italy

joint work with
Roberto Carbone and Luca Compagna

Workshop on Logic and Information Security
Leiden, September 24, 2008
1. Introduction

2. LTL Model Checking for Security Protocol Analysis

3. Analysis of the SAML 2.0 Web Browser SSO Profile

4. Analysis of the ASW Protocol

5. Conclusions
Model checkers specifically tailored for security protocols have been remarkably successful in spotting flaws in protocols.

They rely on a number of simplifying assumptions:

**Perfect Cryptography (PC):** An adversary cannot learn anything from an encrypted message unless he knows the corresponding key.

**Dolev-Yao Intruder (DY):** Communication controlled by a Dolev-Yao (DY) intruder, a malicious agent capable to overhear, divert, and fake messages.

**Honest Principals (HP):** Honest principals are only required to react to messages of a specified form by sending other messages.

**Security Goals (SG):** Security goals are state properties (i.e. invariants).
Model checkers specifically tailored for security protocols have been remarkably successful in spotting flaws in protocols.

They rely on a number of simplifying assumptions:

**Perfect Cryptography (PC):** An adversary cannot learn anything from an encrypted message unless he knows the corresponding key.

**Dolev-Yao Intruder (DY):** Communication controlled by a Dolev-Yao (DY) intruder, a malicious agent capable of overhearing, diverting, and fake messages.

**Honest Principals (HP):** Honest principals are only required to react to messages of a specified form by sending other messages.

**Security Goals (SG):** Security goals are state properties (i.e. invariants).
Model checkers specifically tailored for security protocols have been remarkably successful in spotting flaws in protocols.

They rely on a number of simplifying assumptions:

**Perfect Cryptography (PC):** An adversary cannot learn anything from an encrypted message unless he knows the corresponding key.

**Dolev-Yao Intruder (DY):** Communication controlled by a Dolev-Yao (DY) intruder, a malicious agent capable to overhear, divert, and fake messages.

**Honest Principals (HP):** Honest principals are only required to react to messages of a specified form by sending other messages.

**Security Goals (SG):** Security goals are state properties (i.e. invariants).
Model checkers specifically tailored for security protocols have been remarkably successful in spotting flaws in protocols.

They rely on a number of simplifying assumptions:

**Perfect Cryptography (PC):** An adversary cannot learn anything from an encrypted message unless he knows the corresponding key.

**Dolev-Yao Intruder (DY):** Communication controlled by a Dolev-Yao (DY) intruder, a malicious agent capable to overhear, divert, and fake messages.

**Honest Principals (HP):** Honest principals are only required to react to messages of a specified form by sending other messages.

**Security Goals (SG):** Security goals are state properties (i.e. invariants).
Model checkers specifically tailored for security protocols have been remarkably successful in spotting flaws in protocols.

They rely on a number of simplifying assumptions:

**Perfect Cryptography (PC):** An adversary cannot learn anything from an encrypted message unless he knows the corresponding key.

**Dolev-Yao Intruder (DY):** Communication controlled by a Dolev-Yao (DY) intruder, a malicious agent capable to overhear, divert, and fake messages.

**Honest Principals (HP):** Honest principals are only required to react to messages of a specified form by sending other messages.

**Security Goals (SG):** Security goals are state properties (i.e. invariants).
These assumptions are ok for simple protocols, but they prevent (or greatly complicate) the analysis of whole classes of important protocols.
Problems with the Common Assumptions

(PC) Many security protocols (e.g. Diffie-Hellman) use crypto systems enjoying special algebraic properties.

(DY) DY channels are not appropriate to model the behaviour of an attacker in

- **over-the-air protocols** (message interception unfeasible)
- **contract-signing protocols** (confidential, resilient channels)
- **browser-based protocols** (SSL/TLS channels)

(HP) Some protocols assume “non standard” behaviour of honest principals:

- **contract-signing protocols** (participants required to make progress)
- **browser-based protocols** (HTTP-redirect).

(SG) Some security goals cannot be (easily) expressed as reachability properties, e.g. fair exchange.
Problems with the Common Assumptions

(PC) Many security protocols (e.g. Diffie-Hellman) use crypto systems enjoying special algebraic properties.

(DY) DY channels are not appropriate to model the behaviour of an attacker in

- over-the-air protocols (message interception unfeasible)
- contract-signing protocols (confidential, resilient channels)
- browser-based protocols (SSL/TLS channels)

(HP) Some protocols assume “non standard” behaviour of honest principals:

- contract-signing protocols (participants required to make progress)
- browser-based protocols (HTTP-redirect).

(SG) Some security goals cannot be (easily) expressed as reachability properties, e.g. fair exchange.
Problems with the Common Assumptions

(\textbf{PC}) Many security protocols (e.g. Diffie-Hellman) use crypto systems enjoying special algebraic properties.

(\textbf{DY}) DY channels are not appropriate to model the behaviour of an attacker in

- \textit{over-the-air protocols} (message interception unfeasible)
- \textit{contract-signing protocols} (confidential, resilient channels)
- \textit{browser-based protocols} (SSL/TLS channels)

(\textbf{HP}) Some protocols assume “non standard” behaviour of honest principals:

- \textit{contract-signing protocols} (participants required to make progress)
- \textit{browser-based protocols} (HTTP-redirect).

(\textbf{SG}) Some security goals cannot be (easily) expressed as reachability properties, e.g. fair exchange.
Problems with the Common Assumptions

(\textbf{PC}) Many security protocols (e.g. Diffie-Hellman) use crypto systems enjoying special algebraic properties.

(\textbf{DY}) \textit{DY} channels are not appropriate to model the behaviour of an attacker in

- \textbf{over-the-air protocols} (message interception unfeasible)
- \textbf{contract-signing protocols} (confidential, resilient channels)
- \textbf{browser-based protocols} (SSL/TLS channels)

(\textbf{HP}) Some protocols assume “non standard” behaviour of honest principals:

- \textbf{contract-signing protocols} (participants required to make progress)
- \textbf{browser-based protocols} (HTTP-redirect).

(\textbf{SG}) Some security goals cannot be (easily) expressed as reachability properties, e.g. fair exchange.
I this talk I will present an approach to security protocol analysis based on model checking of LTL formulae.

The approach does not rely on (DY), (HP) and (SG).

Effectiveness of the approach assessed against:
- SAML 2.0 Web Browser Single Sign-On Profile
  ⇒ Flaw detected in Google’s SAML-based SSO for Google Apps
- Optimistic Fair Exchange Protocol by Asokan, Shoup, and Waidner
  ⇒ Flaw detected in a version of the protocol “patched” by Mitchell & Shmatikov [SM02].
I this talk I will present an aproach to security protocol analysis based on model checking of LTL formulae.

The approach does not rely on (DY), (HP) and (SG).

Effectiveness of the approach assessed against:

- SAML 2.0 Web Browser Single Sign-On Profile
  ⇒ Flaw detected in Google’s SAML-based SSO for Google Apps

- Optimistic Fair Exchange Protocol by Asokan, Shoup, and Waidner
  ⇒ Flaw detected in a version of the protocol “patched” by Mitchell & Shmatikov [SM02].
1. Introduction

2. LTL Model Checking for Security Protocol Analysis

3. Analysis of the SAML 2.0 Web Browser SSO Profile

4. Analysis of the ASW Protocol

5. Conclusions
$M$: transition system modelling a superset of the behaviours of the honest agents and of the intruder.

$C_I$: LTL formula constraining the behaviours of the intruder.

$C_H$: LTL formula constraining the behaviours of honest principals.

$G$: LTL formula encoding the expected security property.

By LTL we mean propositional LTL with future (i.e. $G$, $F$, $X$) and past (i.e. $H$, $O$, $Y$) operators.
LTL Model Checking for Security Protocol Analysis

\[ M \models ((C_I \land C_H) \Rightarrow G) \]

- **M**: transition system modelling a superset of the behaviours of the honest agents and of the intruder.
- **\(C_I\)**: LTL formula constraining the behaviours of the intruder.
- **\(C_H\)**: LTL formula constraining the behaviours of honest principals.
- **\(G\)**: LTL formula encoding the expected security property.

By LTL we mean propositional LTL with future (i.e. \(G, F, X\)) and past (i.e. \(H, O, Y\)) operators.
LTL Model Checking for Security Protocol Analysis

\[
\begin{align*}
\text{model } M & \models ((C_I \land C_H) \Rightarrow G) \\
\end{align*}
\]

- \(M\): transition system modelling a superset of the behaviours of the honest agents and of the intruder.
- \(C_I\): LTL formula constraining the behaviours of the intruder.
- \(C_H\): LTL formula constraining the behaviours of honest principals.
- \(G\): LTL formula encoding the expected security property.

By LTL we mean propositional LTL with future (i.e. \(G, F, X\)) and past (i.e. \(H, O, Y\)) operators.
\[
\begin{align*}
\text{model } M & \models (C_I \land C_H) \Rightarrow G,
\end{align*}
\]

- **\(M\):** transition system modelling a superset of the behaviours of the honest agents and of the intruder.
- **\(C_I\):** LTL formula constraining the behaviours of the intruder.
- **\(C_H\):** LTL formula constraining the behaviours of honest principals.
- **\(G\):** LTL formula encoding the expected security property.

By LTL we mean propositional LTL with future (i.e. \(G\), \(F\), \(X\)) and past (i.e. \(H\), \(O\), \(Y\)) operators.
LTL Model Checking for Security Protocol Analysis

\[
\text{model } M \models (((C_I \land C_H) \Rightarrow G)\\
\]

- **M**: transition system modelling a superset of the behaviours of the honest agents and of the intruder.
- **C_I**: LTL formula constraining the behaviours of the intruder.
- **C_H**: LTL formula constraining the behaviours of honest principals.
- **G**: LTL formula encoding the expected security property.

By LTL we mean propositional LTL with future (i.e. G, F, X) and past (i.e. H, O, Y) operators.
The Model

\[ M \models (C_I \land C_H) \Rightarrow G \]

Transition system associated with the concurrent execution of a number of sessions of the protocol.

- **States**: sets of facts, i.e. ground atomic formulae
- **Transitions**: rewrite rules define mappings between sets of facts.
## The Model: Facts

<table>
<thead>
<tr>
<th>Fact</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textbf{state}_{Role}(j, a, es, s)</td>
<td>Principal ( a ), playing role ( Role ), is ready to execute step ( j ) in session ( s ) of the protocol.</td>
</tr>
<tr>
<td>\textbf{ak}(a, m)</td>
<td>Principal ( a ) knows message ( m ).</td>
</tr>
<tr>
<td>\textbf{sent}(rs, b, a, m, c)</td>
<td>Principal ( rs ) has sent message ( m ) on channel ( c ) to principal ( a ) pretending to be principal ( b ).</td>
</tr>
<tr>
<td>\textbf{rcvd}(a, b, m, c)</td>
<td>Message ( m ) (supposedly sent by principal ( b )) has been received on channel ( c ) by principal ( a ).</td>
</tr>
</tbody>
</table>

Note: \( \textbf{ik}(m) \) abbreviates \( \textbf{ak}(i, m) \).

### Example (State):

\[
\begin{align*}
\textbf{state}_{Init}(2, a, [ka, ka^{-1}, kb], 1) \cdot \textbf{sent}(a, a, i, \{\langle a, na \rangle\}_{ki}, c) \\
\cdot \textbf{state}_{Resp}(1, b, [kb, kb^{-1}, ka], 1) \cdot \textbf{ik}(ka) \cdot \textbf{ik}(kb)
\end{align*}
\]
The Model: Rules for the Honest Agents

**Message Delivery**

\[
\text{sent}(RS, B, A, M, C) \xrightarrow{\text{receive}(A,B,RS,M,C)} \text{rcvd}(A, B, M, C) \cdot \text{ak}(A, M)
\]

**Message Processing**

\[
\text{rcvd}(A, B, M, C) \cdot \text{state}_{Role}(j, A, es, S) \xrightarrow{\text{send}_j(A,B,B_1,...,S)} \text{sent}(A, A, B_1, M_1, C_1) \cdot \text{state}_{Role}(l, A, es', S)
\]
The Model: Rules for the Intruder

**Interception**

\[ \text{sent}(A, A, B, M, C) \xrightarrow{\text{intercept}(A,B,M,C)} \text{rcvd}(i, A, M, C).\text{ik}(M) \]

**Overhearing**

\[ \text{sent}(A, A, B, M, C) \xrightarrow{\text{overhear}(A,B,M,C)} \text{sent}(A, A, B, M, C). \text{rcvd}(i, A, M, C).\text{ik}(M) \]

**Faking**

\[ \text{ik}(M).\text{ik}(A).\text{ik}(B) \xrightarrow{\text{fake}(A,B,M,C)} \text{sent}(i, A, B, M, C). \text{ik}(M).\text{ik}(A).\text{ik}(B) \]
The Model: Inferential Capabilities of the Agents

\[ \text{ak}(A, M) . \text{ak}(A, K) \xrightarrow{\text{encrypt}(A, K, M)} \text{ak}(A, M) . \text{ak}(A, K) . \text{ak}(A, \{M\}_K) \]

\[ \text{ak}(A, \{M\}_K) . \text{ak}(A, K^{-1}) \xrightarrow{\text{decrypt.puk}(A, K, M)} \text{ak}(A, \{M\}_K) . \text{ak}(A, K^{-1}) . \text{ak}(A, M) \]

\[ \text{ak}(A, \{M\}_{K-1}) . \text{ak}(A, K) \xrightarrow{\text{decrypt.prk}(A, K, M)} \text{ak}(A, \{M\}_{K-1}) . \text{ak}(A, K) . \text{ak}(A, M) \]

\[ \text{ak}(A, M_1) . \text{ak}(A, M_2) \xrightarrow{\text{pairing}(A, M_1, M_2)} \text{ak}(A, M_1) . \text{ak}(A, M_2) . \text{ak}(A, \langle M_1, M_2 \rangle) \]

\[ \text{ak}(A, \langle M_1, M_2 \rangle) \xrightarrow{\text{decompose}(A, M_1, M_2)} \text{ak}(A, \langle M_1, M_2 \rangle) . \text{ak}(A, M_1) . \text{ak}(A, M_2) \]
Constraining the Behaviour of the Intruder

\[ M \models (C_I \land C_H) \Rightarrow G \]

Confidential Channel

A *channel ch is confidential to principal p* iff its output is exclusively accessible to a given receiver p:

\[
\text{confidential}(ch, p) := G \forall (\text{rcvd}(A, B, M, ch) \Rightarrow A = p)
\]

Authentic Channel

A *channel ch is authentic for principal p* iff its input is exclusively accessible to a given sender p:

\[
\text{authentic}(ch, p) := G \forall (\text{sent}(RS, A, B, M, ch) \Rightarrow (A = p \land RS = p))
\]

- Capital letters denote variables.
- \( \forall (\alpha) \) abbreviates the universal closure of \( \alpha \).
- Quantifiers are over finite domains (bounded analysis).
Constraining the Behaviour of Honest Principals

\[ M \models (C_I \land C_H) \Rightarrow G \]

Principal a should not indefinitely wait for an answer

\[ G \forall (\text{state}_R(j, a, \ldots) \Rightarrow \neg \text{state}_R(j, a, \ldots)) \]

Received messages will be eventually processed by principal a

\[ G \forall (\text{rcvd}(a, P, M, C) \Rightarrow \neg \text{rcvd}(a, P, M, C)) \]
Specifying Security Properties

\[ M \models (C_I \land C_H) \Rightarrow G \]

**Authentication**

*b authenticates a on m in session s* iff

\[
\text{authentication}(b, a, m, s) := \begin{align*}
\text{G} \forall (\text{state}_{rb}(\text{final \_step}, b, [a, \ldots, m, \ldots], s) \Rightarrow \exists \text{O state}_{ra}(\text{initial \_step}, a, [b, \ldots, m, \ldots], s))
\end{align*}
\]

**Secrecy**

*Secrecy of m* holds iff the intruder cannot possibly know it:

\[
\text{secret}(m) := \text{G} \neg \text{i}\text{k}(m)
\]
We have developed SATMC, a bounded model checker for security protocols.

SATMC automatically generates a propositional formula whose satisfying assignments (if any) correspond to counterexamples (i.e. execution traces of $M$ that satisfy $C_I$ and $C_H$ and falsify $G$) of length bounded by some integer $k$.

Successful combination of
- SAT-reduction techniques developed for AI-planning [KMS96]
- Bounded model-checking techniques developed for reactive systems [BCCZ99].

Finding attacks (of length $k$) on the protocol therefore boils down to solving propositional satisfiability problems.
Single-Sign-On (SSO) protocols enable companies to establish a federated environment in which clients sign in once and yet are able to access to services offered by different providers.

The OASIS Security Assertion Markup Language (SAML) 2.0 Web Browser SSO Profile is the emerging standard in this context.

We formally specified and model checked

- the protocol corresponding to one of the most applied use case scenario (the SP-Initiated SSO with Redirect/POST Bindings)
- a variant of the protocol implemented by Google and used (until Sep 30, 2008) by Google’s customers (the SAML-based SSO for Google Applications).

These are browser-based protocols whose security critically relies on secure channels established through SSL/TLS.
SAML 2.0 Web Browser SSO Profile

1. C, SP, URI

2. C, IdP, AuthReq(ID, SP), URI

3. Resp(ID, SP, IdP, \{AA\}_{K_{IdP}^{-1}}), URI

4. Resp(ID, SP, IdP, \{AA\}_{K_{IdP}^{-1}}), URI

SAML Authentication Protocol

IdP builds an authentication assertion

AA = AuthAssert(ID, C, IdP, SP)

S2. Resource
SAML 2.0 Web Browser SSO Profile

A1. C, IdP, AuthReq(ID, SP), URI

A2. C, IdP, AuthReq(ID, SP), URI

IdP builds an authentication assertion

\[ AA = \text{AuthAssert}(ID, C, IdP, SP) \]

A3. Resp(ID, SP, IdP, \{AA\}_{K_{idP}}), URI

S2. Resource
SAML 2.0 Web Browser SSO Profile

C \rightarrow IdP \rightarrow SP

A1. C, IdP, AuthReq(ID, SP), URI

A2. C, IdP, AuthReq(ID, SP), URI

A3. Resp(ID, SP, IdP, \{AA\}_{K_{idP}}^{-1}), URI

A4. Resp(ID, SP, IdP, \{AA\}_{K_{idP}}^{-1}), URI

IdP builds an authentication assertion

AA = AuthAssert(ID, C, IdP, SP)

S1. C, SP, URI

S2. Resource
SAML 2.0 Web Browser SSO Profile

SAML Authentication Protocol

A1. \(C, IdP, \text{AuthReq}(ID, SP), \text{URI}\)

A2. \(C, IdP, \text{AuthReq}(ID, SP), \text{URI}\)

A3. \(\text{Resp}(ID, SP, IdP, \{AA\}_{K_{idP}}^{-1}), \text{URI}\)

A4. \(\text{Resp}(ID, SP, IdP, \{AA\}_{K_{idP}}^{-1}), \text{URI}\)

IdP builds an authentication assertion

\(AA = \text{AuthAssert}(ID, C, IdP, SP)\)

S2. Resource
Assumptions & Goals

- **Trust.** The protocol assumes that IdP is trustworthy for both C and SP, but neither SP nor C are assumed to be trustworthy.

- **Channels.**
  - (A1) communication between C and SP is carried over a unilateral SSL/TLS
  - (A2) communication between C and IdP is carried over a unilateral SSL/TLS channel that becomes bilateral once C authenticates itself on IdP.

- **Goals.**
  - SP authenticates C, i.e., at the end of its protocol run SP believes it has been talking with C.
  - Resource must be kept secret between C and SP.
Google Apps

Web applications for communication and collaboration

Stay connected and be more productive

For personal use
Keep in touch and share with friends and family. Free, intuitive tools you can access anywhere with a single account.

- **Gmail**
  Fast, searchable email with less spam

- **Google Talk**
  IM and call your friends through your computer

- **Google Calendar**
  Organize your schedule and share events with friends

- **Google Docs**
  Share online documents, presentations, and spreadsheets

- **Google Sites**
  Create websites and secure group wikis

And much more...

For businesses and schools
Put Google's web-based communication, collaboration and security apps to work for your company or school.

- **Business IT managers**
  Not an IT manager?
  Start collaborating with coworkers or classmates.

- **School IT managers**

©2009 Google
Terms of Service
Program Policies
Help
The “SAML-based” SSO for Google Apps

Same as the SAML 2.0 Web Browser SSO Profile except for a seemingly minor simplifications:

1. ID and SP are not included in the authentication assertion, i.e. AA is AuthAssert(C, IdP) instead of AuthAssert(ID, C, IdP, SP);
2. ID, SP and IdP are not included in the response, i.e. Resp is Resp(\{AA\}_{K_{IdP}}^{-1}) instead of Resp(ID, SP, IdP, \{AA\}_{K_{IdP}}^{-1}).
Attack on the SAML-based SSO for Google Apps

C

IdP

SP

A1. \textit{ailab}, \textit{authReq}(id_i, i), \textit{uri} \\
\textit{bob2i}

A2. \textit{ailab}, \textit{authReq}(id_i, i), \textit{uri} \\
\textit{bob2ailab}

A3. \textit{i}, \textit{response}({AA}_{i-1}^{ailab}), \textit{uri} \\
\textit{ailab2bob}

A4. \textit{i}, \textit{response}({AA}_{i-1}^{ailab}), \textit{uri} \\
\textit{bob2i}

A1. \textit{ailab}, \textit{authReq}(id_{google}, google), \textit{calendar} \\
\textit{i2google}

A1. \textit{ailab}, \textit{authReq}(id_{google}, google), \textit{calendar} \\
\textit{google2i}

\textit{ailab} builds an authentication assertion \( AA = \text{authAssert}(bob, ailab) \)

A4. \textit{google}, \textit{response}({AA}_{i-1}^{ailab}), \textit{calendar} \\
\textit{i2google}

S2. resource \\
\textit{google2i}
Attack on the SAML-based SSO for Google Apps

C

IdP

SP

C

S1. bob, i, uri

A1. ailab, authReq(id_i, i), uri

A2. ailab, authReq(id_i, i), uri

A3. i, response({AA}^{k−1}_{ailab}), uri

A4. i, response({AA}^{k−1}_{ailab}, uri

S1. bob, google, calendar

A1. ailab, authReq(id_{google}, google), calendar

A2. google2i

A3. i2i

A4. google, response({AA}^{k−1}_{ailab}), calendar

A2. bob2i

A4. bob2i

ailab builds an authentication assertion AA = authAssert(bob, ailab)

S2. resource

Alessandro Armando (U. Genova)
We have reproduced the above attack in an actual deployment of the SAML-based SSO for Google Apps:

1. 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.
2. We then implemented a Java Servlet simulating the behaviour of a dishonest SP called BadSP.
3. After 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.
4. 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.
5. The attack succeeds and Google logs BadSP into Google Apps as the AI-Lab member.
Vulnerability Note VU#612636

Google SAML Single Sign on vulnerability

Overview

The SAML Single Sign-On (SSO) Service for Google Apps contained a vulnerability that could have allowed an attacker to gain access to a user's Google account.

I. Description

The Security Assertion Markup Language (SAML) is a standard for transmitting authentication data between two or more security domains. In SAML language, XML security packets are called assertions. Identity providers pass assertions to service providers who allow the requests. In the Google Single Sign on (SSO) implementation, the authentication response did not include the identifier of the authentication request or the identity of the recipient. This may allow a malicious service provider to impersonate a user at other service providers.

More technical information about this issue is available in the Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps whitepaper which is available here: http://www.al-lab.it/armando/GoogleSSOVulnerability.html

Note that to exploit this vulnerability, the attacker would have to convince the user to login to their site.

II. Impact
In June we made a change in SSO requirements:
https://groups.google.com/group/google-apps-apis/browse_thread/thread/8183040d7980a2e6

It was in response to an issue which security researchers reported to us about the Google Apps SAML service provider.

Now that the details of the security issue are published:
http://www.kb.cert.org/vuls/id/612636

we would like to thank Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, and Llanos Tobarra Abad with the AVANTSSAR project for responsible disclosure of this vulnerability.

Their paper which describes using a model checker on the SAML protocol implementation to uncover a vulnerability is here:

-aless
Details can be found in


Paper, links and a video demonstrating the attack available at

http://www.ai-lab.it/armando/GoogleSSOVulnerability.html
Outline

1. Introduction
2. LTL Model Checking for Security Protocol Analysis
3. Analysis of the SAML 2.0 Web Browser SSO Profile
4. Analysis of the ASW Protocol
5. Conclusions
The ASW Protocol

The ASW protocol consists of three subprotocols.

Exchange: played by $O$ and $R$ and—if successful—it allows to achieve a mutual commitment on a previously agreed contractual text (standard contract).

Abort: $O$ can request $T$ to abort the previously initiated contract signing procedure.

Resolve: $O$ and $R$ can request $T$ to force the resolution of the contract, possibly obtaining a replacement contract.
The ASW Protocol

The ASW protocol consists of three subprotocols.

**Exchange:** played by $O$ and $R$ and—if successful—it allows to achieve a mutual commitment on a previously agreed contractual text (standard contract).

**Abort:** $O$ can request $T$ to abort the previously initiated contract signing procedure.

**Resolve:** $O$ and $R$ can request $T$ to force the resolution of the contract, possibly obtaining a replacement contract.
The ASW Protocol

The ASW protocol consists of three subprotocols.

**Exchange:** played by $O$ and $R$ and—if successful—it allows to achieve a mutual commitment on a previously agreed contractual text (standard contract).

**Abort:** $O$ can request $T$ to abort the previously initiated contract signing procedure.

**Resolve:** $O$ and $R$ can request $T$ to force the resolution of the contract, possibly obtaining a replacement contract.
The Exchange Subprotocol

1st round: $O$ and $R$ express their **public commitments** to the contract.
2nd round: $O$ and $R$ exchange their **secret commitments**, required for the **standard contract** (i.e. \{ $me_1$, $N_O$, $me_2$, $N_R$ \}).

\begin{align*}
E1. \quad O \rightarrow R & : \quad me_1 = \text{Sig}_O(V_O, V_R, T, \text{Text}, h(N_O)) \\
E2. \quad R \rightarrow O & : \quad me_2 = \text{Sig}_R(me_1, h(N_R)) \\
E3. \quad O \rightarrow R & : \quad N_O \\
E4. \quad R \rightarrow O & : \quad N_R
\end{align*}
The Exchange Subprotocol

1st round: $O$ and $R$ express their public commitments to the contract. 

2nd round: $O$ and $R$ exchange their secret commitments, required for the standard contract (i.e. $\{ me_1, N_O, me_2, N_R \}$).

\[ E1. \quad O \to R \quad : \quad me_1 = \text{Sig}_O(V_O, V_R, T, \text{Text}, h(N_O)) \]

\[ E2. \quad R \to O \quad : \quad me_2 = \text{Sig}_R(me_1, h(N_R)) \]

\[ E3. \quad O \to R \quad : \quad N_O \]

\[ E4. \quad R \to O \quad : \quad N_R \]
The Abort Subprotocol

A1. \( O \to T \) : \( ma_1 = \text{Sig}_O(\text{aborted}, me_1) \)

A2. \( T \to O \) : if resolved\((me_1, me_2) \in DB\) 
then \( \text{Sig}_T(me_1, me_2) \)
// Replacement contract
else \( DB := DB \cup \{\text{aborted}(me_1)\}; \text{Sig}_T(\text{aborted}, ma_1) \)
// Abort token

Timeout for O
The Abort Subprotocol

A1.  \( O \rightarrow T \) : \( ma_1 = Sig_O(aborted, me_1) \)

A2.  \( T \rightarrow O \) : if \( \text{resolved}(me_1, me_2) \in DB \) then \( Sig_T(me_1, me_2) \)

\[ \text{// Replacement contract} \]

else \( DB := DB \cup \{ \text{aborted}(me_1) \} \);

\( Sig_T(aborted, ma_1) \)

\[ \text{// Abort token} \]
The Abort Subprotocol

A1. $O \rightarrow T : \quad ma_1 = \text{Sig}_O(\text{aborted}, me_1)$

A2. $T \rightarrow O : \quad \text{if resolved}(me_1, me_2) \in DB$
then $\text{Sig}_T(me_1, me_2)$
  // Replacement contract
else $DB := DB \cup \{\text{aborted}(me_1)\}$;
  $\text{Sig}_T(\text{aborted}, ma_1)$
  // Abort token
The Resolve Subprotocol (1)

\[ R \rightarrow T : mr_1 = \langle me_1, me_2 \rangle \]

\[ T \rightarrow R : \begin{cases} 
\text{if } \text{aborted}(me_1) \in DB \\
\text{then } \text{Sig}_T(\text{aborted}, ma_1) \\
\text{// Abort token} \\
\text{else } DB := DB \cup \{\text{resolved}(me_1, me_2)\}; \\
\text{Sig}_T(me_1, me_2) \\
\text{// Replacement contract} 
\end{cases} \]
The Resolve Subprotocol (1)

\[ R_1. \ R \rightarrow T : \ mr_1 = \langle me_1, me_2 \rangle \]

\[ R_2. \ T \rightarrow R : \begin{align*} &\text{if } \text{aborted}(me_1) \in DB \text{ then } \Sigma_T(aborted, ma_1) \\
&\quad \quad \quad \quad \quad \text{// Abort token} \\
&\text{else } DB := DB \cup \{\text{resolved}(me_1, me_2)\}; \\
&\quad \quad \quad \quad \quad \Sigma_T(me_1, me_2) \\
&\quad \quad \quad \quad \quad \text{// Replacement contract} \end{align*} \]
R1. \( R \rightarrow T \) : \( mr_1 = \langle me_1, me_2 \rangle \)

R2. \( T \rightarrow R \) : if aborted\( (me_1) \) \( \in \) \( DB \)  
\hspace{1cm} then \( \text{Sig}_T(aborted, ma_1) \)  
\hspace{1cm} // Abort token  
\hspace{1cm} else \( DB := DB \cup \{\text{resolved}(me_1, me_2)\} \);  
\hspace{1cm} \( \text{Sig}_T(me_1, me_2) \)  
\hspace{1cm} // Replacement contract
The Resolve Subprotocol (2)

R1. $O \rightarrow T : mr_1 = \langle me_1, me_2 \rangle$

R2. $T \rightarrow O :$ if aborted$(me_1) \in DB$
then $\text{Sig}_T(\text{aborted}, ma_1)$
  // Abort token
else $DB := DB \cup \{\text{resolved}(me_1, me_2)\}$;
$\text{Sig}_T(me_1, me_2)$
  // Replacement contract
- **Confidential**: eavesdroppers do not have access to the information.

- **Resilient**: any message will be eventually delivered to the intended recipient.
**Confidential**: eavesdroppers do not have access to the information.

**Resilient**: any message will be eventually delivered to the intended recipient.
**Timeout** ($O$ and $R$): During the exchange sub-protocol, $O$ and $R$ will not indefinitely wait for a reply from the corresponding party and will eventually start either the abort or resolve subprotocol.

**Availability** ($T$): $T$ must be always available, i.e. he must eventually process all the messages received by replying to requests by $O$ and $R$. 
Fair exchange can be expressed as the conjunction of the following two properties:

(A) A principal cannot obtain a valid contract without allowing the remaining principal to also obtain a valid contract.

(B) Once a principal obtains an abort token, it is impossible for any other principal to obtain a valid contract.
Fair Exchange (A): Specification

\[ M \models (C_I \land C_H) \Rightarrow G_A \]

(A) A principal cannot obtain a valid contract without allowing the remaining principal to also obtain a valid contract.

\[ G_A \] is the conjunction, for each session considered, of

\[ \mathbf{G}\forall(\text{hasVC}(o, o, r, \text{txt}, N_O, N_R, t) \Rightarrow \mathbf{F}\text{hasVC}(r, o, r, \text{txt}, N_O, N_R, t)) \]

Informally: if \( o \) has a valid contract, binding \( r \) to \( \text{txt} \) using \( N_O \) and \( N_R \) as secret commitments and \( t \) as \( T \), then eventually \( r \) will have a corresponding valid contract.
hasVC(P, O, R, Txt, NO, NR, T) :=

// P has a standard contract, i.e. he knows me1, me2, NO, and NR
[ ak(P, me1(O, R, Txt, NO, T)) ∧
ak(P, me2(O, R, Txt, NO, NR, T)) ∧
ak(P, NO) ∧
ak(P, NR) ]

∨

// P has a replacement contract, i.e. he knows SigT(me1, me2)
 ak(P, SigT(me1(O, R, Txt, NO, T), me2(O, R, Txt, NO, NR, T)))
SATMC finds the following attack:

\[
\begin{align*}
E1. & \quad O \to I : \quad me_1 \\
E2. & \quad I \to O : \quad me_2 \\
& \quad I \text{ computes new random } N'_I \text{ and then } me'_2 \\
E3. & \quad O \to I : \quad NO \\
E4. & \quad I \to O : \quad NI
\end{align*}
\]

At the end of the protocol, the intruder owns a standard contract \(\{me_1, me'_2, NO, N'_I\}\) while \(O\) owns only the standard contract \(\{me_1, me_2, NO, NI\}\).

This attack is similar in spirit to the attack in [SM02], but it is simpler as \(T\) is not involved.
SATMC finds the following attack:

\begin{align*}
E1. \quad O &\rightarrow I \quad : \quad me_1 \\
E2. \quad I &\rightarrow O \quad : \quad me_2 \\
& \quad I \text{ computes new random } N'_I \text{ and then } me'_2 \\
E3. \quad O &\rightarrow I \quad : \quad NO \\
E4. \quad I &\rightarrow O \quad : \quad NI
\end{align*}

At the end of the protocol, the intruder owns a standard contract 
\( \{me_1, me'_2, NO, N'_I\} \) while \( O \) owns only the standard contract 
\( \{me_1, me_2, NO, NI\} \).

This attack is similar in spirit to the attack in [SM02], but it is simpler as \( T \) is not involved.
Mitchell & Shmatikov propose to repair ASW by changing $E_3$ and $E_4$:

$$E_3. \quad O \rightarrow R : \quad N_O \quad \Rightarrow \quad E_3'. \quad O \rightarrow R : \quad \text{Sig}_O(N_O, h(N_R))$$

$$E_4. \quad R \rightarrow O : \quad N_R \quad \Rightarrow \quad E_4'. \quad R \rightarrow O : \quad \text{Sig}_R(N_R, h(N_O))$$

SATMC confirms that the improved version of the protocol does not suffer from the previous attack, but detects a new (i.e. previously unknown) attack!

$$E_1. \quad O \rightarrow I : \quad me_1$$

$$E_2. \quad I \rightarrow O : \quad me_2$$

$I$ computes new random $N'_I$ and then $me'_2$

$$E_3'. \quad O \rightarrow I : \quad \text{Sig}_O(N_O, h(N_I))$$

$$E_4'. \quad I \rightarrow O : \quad \text{Sig}_I(N_I, h(N_O))$$

$$R1. \quad I \rightarrow T : \quad mr_1 = \langle me_1, me'_2 \rangle$$

$$R2. \quad T \rightarrow I : \quad mr_2 = \text{Sig}_T(me_1, me'_2)$$

The intruder, here playing the responder, obtains a repl. contract relative to $N_O$ and $N'_I$, while $O$ obtains only a valid contract relative to $N_O$ and $N_I$. 
Mitchell & Shmatikov propose to repair ASW by changing $E_3$ and $E_4$:

$$E_3. \quad O \rightarrow R : \quad N_O \quad \quad \Rightarrow \quad \quad E_3'. \quad O \rightarrow R : \quad \text{Sig}_O(N_O, h(N_R))$$

$$E_4. \quad R \rightarrow O : \quad N_R \quad \quad \Rightarrow \quad \quad E_4'. \quad R \rightarrow O : \quad \text{Sig}_R(N_R, h(N_O))$$

SATMC confirms that the improved version of the protocol does not suffer from the previous attack, but detects a new (i.e. previously unknown) attack!

$$E_1. \quad O \rightarrow I : \quad \text{me}_1$$

$$E_2. \quad I \rightarrow O : \quad \text{me}_2$$

$I$ computes new random $N'_I$ and then $\text{me}'_2$

$$E_3'. \quad O \rightarrow I : \quad \text{Sig}_O(N_O, h(N_I))$$

$$E_4'. \quad I \rightarrow O : \quad \text{Sig}_I(N_I, h(N_O))$$

$$R1. \quad I \rightarrow T : \quad \text{mr}_1 = \langle \text{me}_1, \text{me}'_2 \rangle$$

$$R2. \quad T \rightarrow I : \quad \text{mr}_2 = \text{Sig}_T(\text{me}_1, \text{me}'_2)$$

The intruder, here playing the responder, obtains a repl. contract relative to $N_O$ and $N'_I$, while $O$ obtains only a valid contract relative to $N_O$ and $N_I$. 
Also this new attack does not have serious consequences: \( O \) and \( R \) do not have the same contract, but they have a contract for the same text.

Yet the ability to detect this attack, which has eluded previous formal analyses of the protocol, gives evidence of the effectiveness of the approach.

We have come to the conclusion that the problem is not in the protocol but in the property, which requires the contract to be relative to the same secret commitment \( N_R \).

We have therefore weakened the property accordingly and successfully checked it with SATMC.

\[
\mathbf{G} \forall(\text{hasVC}(o, o, r, \text{txt}, N_O, N_R, t)) \Rightarrow \mathbf{F} \text{hasVC}(r, o, r, \text{txt}, N_O, N_R, t)
\]
Also this new attack **does not have serious consequences**: \( O \) and \( R \) do not have the same contract, but they have a contract for the same text.

Yet the ability to detect this attack, which has eluded previous formal analyses of the protocol, gives evidence of the **effectiveness** of the approach.

We have come to the conclusion that the problem is not in the protocol but in the property, which requires the contract to be relative to the same secret commitment \( N_R \).

We have therefore weakened the property accordingly and successfully checked it with SATMC.

\[
\forall (\text{hasVC}(o, o, r, txt, N_O, N_R, t) \Rightarrow F \exists N'_R. \text{hasVC}(r, o, r, txt, N_O, N'_R, t))
\]
Details on the approach and on the analysis of the ASW protocol can be found in


Outline

1. Introduction
2. LTL Model Checking for Security Protocol Analysis
3. Analysis of the SAML 2.0 Web Browser SSO Profile
4. Analysis of the ASW Protocol
5. Conclusions
Conclusions

Most state-of-the-art security protocol analysers cannot tackle interesting classes of protocols (browser based protocols, contract signing protocols, ...)

LTL improves considerably the scope of model checking for security protocols as it support the specification of:
  - secure channels
  - non standard behaviour of honest principals
  - complex security goals

It works! Vulnerabilities detected on two important protocols:
  - Google’s SAML-based SSO for Google Apps
  - Original and “patched” version of the ASW protocol
Conclusions

Most state-of-the-art security protocol analysers cannot tackle interesting classes of protocols (browser based protocols, contract signing protocols, ...)

LTL improves considerably the scope of model checking for security protocols as it support the specification of:

- secure channels
- non standard behaviour of honest principals
- complex security goals

It works! Vulnerabilities detected on two important protocols:

- Google’s SAML-based SSO for Google Apps
- Original and “patched” version of the ASW protocol
Conclusions

- Most state-of-the-art security protocol analysers cannot tackle interesting classes of protocols (browser based protocols, contract signing protocols, . . . )
- LTL improves considerably the scope of model checking for security protocols as it support the specification of:
  - secure channels
  - non standard behaviour of honest principals
  - complex security goals
- **It works!** Vulnerabilities detected on two important protocols:
  - Google’s SAML-based SSO for Google Apps
  - Original and “patched” version of the ASW protocol
Acknowledgements

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).
A. Biere, A. Cimatti, E. Clarke, and Y. Zhu.
Symbolic Model Checking without BDDs.
Springer-Verlag, 1999.

H. Kautz, H. McAllester, and B. Selman.
Encoding Plans in Propositional Logic.

V. Shmatikov and J. C. Mitchell.
Finite-state analysis of two contract signing protocols.