Analysis Of SET In CPAL-ES

Presented By : Raja Ramaswamy
Date : 04/22/2002
Advisor : Dr. Alec Yasinsac

Organization of the presentation . . .

• Project Goal
• Problems in E-Commerce
• CPAL - ES environment
• SET - An Introduction
• Analysis of SET using CPAL - ES
• Conclusion
**Project Goal**


---

**Problems in e-commerce... Intruder & Imposter**
Solving Intrusion

Cryptography

Symmetric Key, Asymmetric Key

→ Symmetric Key Cryptography

\[
\begin{align*}
\text{Entity}_1 \rightarrow \text{message}^K \rightarrow \text{Entity}_2
\end{align*}
\]

Decodes message using key \(K\)

Example: Data Encryption Standard (DES)

* 56 bit long key, both party should know the keys

CPAL – ES

(Cryptographic Protocol Analysis Language - Evaluation System)

• A formal cryptographic protocol evaluation system.
• It is based on a technique from program verification called Weakest Precondition reasoning.
• Allows analyst to give a definitive meaning to the actions of all principals in a protocol run, including intruders.
• Translates syntax to semantics and declarative result that is provable.
Evaluation of protocols in CPAL-ES is a **three-step process**

1. Encode the protocol actions in CPAL
2. Translate the specification into a Verification Condition
3. Prove the Verification Condition.

**CPAL** (Cryptographic Protocol Analysis Language)

- expressiveness to enable protocol designers to specify complex protocols
- allows analysis of coded protocols through formal methods
- superset of the de facto standard notation (SN)
- simple language

- Some **basic features** include
  - specifying **actions of principals** in a communication session
  - Sending messages
  - encrypting values
  - creating **new** values
**CPAL**

**Advanced features of CPAL**

- every CPAL action preceded by *identifier of the principal*
- every action in the protocol can be *bound to a specific principal*
- requires *receipt* of messages
- explicit *decryption* of encrypted messages
- *dot notation* is used to identify the address space (ex A.k)
- allowing principals to encode protocol goals and assumptions directly into the specification (*assume & assert*)

### Examples:

<table>
<thead>
<tr>
<th>SN Code</th>
<th>CPAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>A -&gt; B {msg}k</td>
<td>A: =&gt;B(e[msg]k);</td>
</tr>
<tr>
<td></td>
<td>B: &lt;-msg';</td>
</tr>
<tr>
<td></td>
<td>B: msg := d[msg']k;</td>
</tr>
</tbody>
</table>

**Assume & Assert Statements**

A: assume(A.k == B.k);
A: assert(A.Na == A.Na');
### SN and CPAL Specification of a Trivial Protocol

<table>
<thead>
<tr>
<th>SN</th>
<th>CPAL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SN</strong></td>
<td><strong>CPAL</strong></td>
</tr>
<tr>
<td>global: assume (A.k==B.k);</td>
<td></td>
</tr>
<tr>
<td><strong>A-&gt;B</strong>: (A)</td>
<td><strong>A</strong>: -&gt; B(A);</td>
</tr>
<tr>
<td><strong>B</strong>: &lt;- (A);</td>
<td></td>
</tr>
<tr>
<td><strong>B-&gt;A</strong>: {N}^k</td>
<td><strong>B</strong>: -&gt; A(e[n]k);</td>
</tr>
<tr>
<td><strong>A</strong>: &lt;- (msg);</td>
<td><strong>A</strong>: n := d[msg]k;</td>
</tr>
<tr>
<td><strong>A</strong>: n := d[msg]k;</td>
<td><strong>B</strong>: &lt;- (msg);</td>
</tr>
<tr>
<td><strong>A</strong>: n' := d[msg]k;</td>
<td><strong>B</strong>: n' := d[msg]k;</td>
</tr>
<tr>
<td><strong>B</strong>: assert (n' == f(n));</td>
<td></td>
</tr>
</tbody>
</table>

### SET Fundamentals

#### Digital Certificates
- Account Information
- Information about the certificate
- Cryptographic Information (Public key of the customer)
- It's an electronic representation of the card

#### Trust Chaining
- Root Certifying Authority (RCA)
- Certifying Authority (CA)
- Customer

**Digital certificates bind the entity with their public key**
What is SET (Secure Electronic Transactions)?

- A modern protocol designed to facilitate secure payment transactions in open networks (ex. Internet)
- This open protocol is jointly developed by MasterCard and Visa with support from IBM, Microsoft, GTE, VeriSign, etc.
- To be used mainly in the E-Commerce arena

How SET works?

SET uses encryption technology and digital certificates as the basis for security and authentication.

Several components to make SET work are:

- Certificates
- Cardholder Wallet and Encryption
- Merchant website and server software
- Payment Gateway
**Cardholder Wallet and Cryptography**

- A software that resides in the customers browser
- Contains payment card information such as
  - Card number
  - Digital certificate to identify the user
  - Shipping information
- Performs the necessary encryption of data
- Most recent browser versions from Microsoft and Netscape support wallet technology

**Merchant website and server software**

- A virtual storefront
- A web server
- Software modules such as secure payment, order management, online customer service etc.
- Encryption and decryption
Payment Gateway

- Interface between the merchant and the acquirers payment processing system
- Validates both merchant and cardholder certificates
- Translate the SET message into a format that can be processed by the Acquirers system

Customer Side activities

Step I

Customer \(\{\text{message}\}\) \(\rightarrow\) Message Digest \(\rightarrow\) Digital Signature

Step II

\(\{\text{message}\}\) + Digital Signature + Certificate \(\rightarrow\) Random Symmetric Key \(\rightarrow\) Encrypted Message

Step III

Random Symmetric Key \(\rightarrow\) Merchants public key \(\rightarrow\) Digital Envelope

Finally

Encrypted Message + Digital Envelope \(\rightarrow\) Merchant
Merchant Side activities

Digital Envelope → Merchants private key → Random Symmetric Key

Encrypted Message → Random Symmetric Key

Digital Signature → Customer Public key → Message Digest

Message Digest + Digital Signature + Certificate

Proceed to Compare

Customer - Bank Transaction

Customer → Merchant → Issuer → Acquirer

Begin Transaction

Confirm Sale

Confirm Payment

Request Funds

Request Authorization

Authorize payment
CPAL Specification for SET

→ Sequential flow of messages
→ Identify the flow of messages between different entities
→ Categorize them
→ Convert them to CPAL
→ Analyze CPAL specification in CPAL-ES

Categories of SET messages

1. Certificate Management messages
2. Cardholder-Merchant messages
3. Merchant-Payment Gateway messages
Cardholder-Merchant messages

* Purchase Initialization Request (PInitReq)
* Purchase Initialization Response (PInitRes)
* Purchase Request (PReq) & Payment Response (PRes)

Purchase Initialization Request (PInitReq) – Generation Process

1. Generate RRPID for matching the message and the matching response message.
2. Populate language of the cardholder’s choice.
3. Generate LID_C, Local Identification for Cardholder
4. If Merchant has already supplied a LID_M in the SET initiation process then copy it into the message.
5. Generate a fresh Chall_C
6. Populate BIN (first 6 digits of the cardholder’s account number)
7. Save RRPID, LID_C, LID_M (if available) and Chall_C
8. Invoke Compose Message Wrapper to send the message to Merchant.
CPAL specification of the PInitReq message

1. Action : Initiate Transaction
2. Message : Purchase Initialization Request (PInitReq)
3. Initiated by : Customer
4. C: RRPID := new;
5. C: Chall_C := new;
7. C: TransRec := <RRPID, LID_C, LID_M, Chall_C>
8. C: MWCAInitReq := <Version, Revision, Date, PInitReq, XID>
9. C: => M (MWCAInitReq);
10. M: <- (MWCAInitReq);

Assertions

Encode protocol goals and assumptions directly into the specification

C: assume (C.RRPID_MWREQ == C.RRPID_REQ);
C: RRPID := new;
C: LID_E := new;
C: Chall_EE := new;
C: CardCInitReq := <RRPID, LID_EE, Chall_EE, BrandID>
C: MWCardCInitReq := <Version, Revision, Date, RRPID, SWIdent, CardCInitReq, XID>
C: => CCA(MWCardCInitReq);
CCA: <- (MWCardCInitReq);
CCA: (Version, Revision, Date, RRPID, SWIdent, CardCInitReq, XID) := MWCardCInitReq;
CCA: (RRPID_MW, LID_EE, Chall_EE, BrandID) := CardCInitReq;
CCA: TransRec := <RRPID, LID_EE, Chall_EE, BrandID>
CCA: assert (RRPID_MW == RRPID_MW)
**Encoded CPAL file**

- `cpal-set.cpa` contains the CPAL specification of the SET protocol

**Generated Files**

- `protocol.out` contains the encoded protocol, Initial WP predicate, Basic simplification and the simplified predicate
- `assume.out` contains assume statements
- `tst.out` contains information to debug

---

**Initial Predicate list**

```
****** Initial WP predicate follows.

| ((true) (true, v12)) =>
| . . . . . .
| . . . . . .

| false (true, v12) =>
| . . . . . .
| . . . . . .
```

**Final Simplified Predicate**

```
****** Simplified predicate follows.

| TRUE |

****** NO MORE PREDICATE
```
**Demonstration** …

- Modules in CPAL file
- Assume and Assert statements
- Final simplified predicate

**Conclusion**

- Learn SET protocol and CPAL
- Convert the SET protocol into a CPAL specification
- Analyze the CPAL specification against the CPAL-ES environment

Thank You & Questions??