Towards Verification of the PANA Authentication and Authorisation Protocol using Coloured Petri Nets

Steven Gordon

SIIT, Thammasat University
Thailand
Outline

• What are EAP and PANA?
• Modelling PANA with CPNs: Why? How?
• Analysing PANA
  - Are there any deadlocks?
  - What are the effects of retransmissions?
  - How does PANA interact with users (EAP)?
• Observations and future work
EAP and PANA
Authentication using EAP

- Extensible Authentication Protocol (EAP)
  - Designed for network access authentication when IP access is not available
    - PPP (dialup), IEEE 802.11 wireless LANs
  - Request/Response messages exchanged between authenticator and peer
EAP Messages

Peer responds depending on type of request: Identity-Request, MD5-Challenge, ...

Authenticator initiates with an Authentication Request

Requests continue until Authenticator has sufficient information from Peer

Success or Failure message sent depending on responses from Peer
PANA and EAP

- Protocol for carrying Authentication for Network Access (PANA)
  - Designed to carry EAP messages over IP networks
Development of PANA by IETF

- **EAP**
  - Protocol description: IETF RFC 3478, June 2004

- **PANA**
    - Started 2003, 18 versions
  - State machines: IETF RFC 5609, Aug 2009
    - Started 2005, 13 versions
  - (CPN modelling and analysis of PANA
    - Started 2007, total of 3-4 months of effort)
PANA Operation

• PANA messages are used to:
  - Initialise and maintain the EAP session
  - Carry EAP messages (Request, Response, Success, Failure) between Peer and Authenticator

• PANA maintains its own session. Four phases:
  1) Authentication and Authorisation: perform the EAP authentication
  2) Access: once authentication, maintain the session
  3) Re-authentication: if session is about to expire
  4) Termination: either PaC or PAA may terminate
PANA Authentication Message Sequence Chart (No Piggybacking) – Part 1

PaC: PANA Client
PAA: PANA Authenticator
PANA Authentication Message Sequence Chart (No Piggybacking) – Part 2
Piggybacking Off

- EAP_REQUEST
- AuthRequest(9)
- AuthAnswer(9)
- AuthRequest(3)
- AuthAnswer(3)
- AuthRequest(C,10)
- AuthAnswer(C,10)
- EAP_SUCCESS

Piggybacking On

- EAP_REQUEST
- AuthRequest(9)
- EAP_RESPONSE
- EAPResponse carried in AuthAnswer
- AuthAnswer(9)
- AuthAnswer(9,EAP)
- AuthRequest(C,10)
- EAP_SUCCESS

- Ack carried in AuthRequest
- AuthRequest(C,10)
- EAP_SUCCESS
- AuthAnswer(C,10)
- AuthAnswer(C,10)
Other PANA Features

- Initiation of PANA session
  - By either PaC or PAA (method out of scope of PANA)
  - Optimised Initiation: if On, PAA can send EAP Request in initial AuthRequest message

- Each Request contains a 32-bit sequence number; responding Answer contains the same number

- Requests can be retransmitted if no answer; abort session if too many retransmits

- PANA messages use Attribute Value Pairs to carry EAP messages, authentication data and other security information
EAP/PANA Interface

- Protocols may define service offered to higher layer (e.g. EAP) independent from protocol operation
  - What messages can be exchanged between PANA and EAP, and in what order?
    - No explicit definition in PANA RFCs
    - Extract information from PANA RFC and EAP RFC: describe variables for communication between PANA/EAP
# EAP/PANA Interface

<table>
<thead>
<tr>
<th>No.</th>
<th>Entity</th>
<th>EAP</th>
<th>PANA</th>
<th>Primitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Peer/PAC</td>
<td>-</td>
<td>AUTH_USER</td>
<td>CAuthUser</td>
</tr>
<tr>
<td>2</td>
<td>Peer/PAC</td>
<td>eapRestart</td>
<td>EAP_RESTART</td>
<td>CRestart</td>
</tr>
<tr>
<td>3</td>
<td>Peer/PaC</td>
<td>eapReq</td>
<td>EAP_REQUEST</td>
<td>CRequest</td>
</tr>
<tr>
<td>4</td>
<td>Peer/PaC</td>
<td>eapResp</td>
<td>EAP_RESPONSE</td>
<td>CResponse</td>
</tr>
<tr>
<td>5</td>
<td>Peer/PaC</td>
<td>eapSuccess</td>
<td>EAP_SUCCESS</td>
<td>CSuccess</td>
</tr>
<tr>
<td>6</td>
<td>Peer/PaC</td>
<td>eapFail</td>
<td>EAP_FAILURE</td>
<td>CFailure</td>
</tr>
<tr>
<td>7</td>
<td>Peer/PaC</td>
<td>-</td>
<td>-</td>
<td>CTimeout</td>
</tr>
<tr>
<td>8</td>
<td>Peer/PaC</td>
<td>-</td>
<td>ABORT</td>
<td>CAbort</td>
</tr>
<tr>
<td>9</td>
<td>Auth/PAA</td>
<td>-</td>
<td>PAC_FOUND</td>
<td>APacFound</td>
</tr>
<tr>
<td>10</td>
<td>Auth/PAA</td>
<td>eapRestart</td>
<td>EAP_RESTART</td>
<td>ARestart</td>
</tr>
<tr>
<td>11</td>
<td>Auth/PAA</td>
<td>eapReq</td>
<td>EAP_REQUEST</td>
<td>ARequest</td>
</tr>
<tr>
<td>12</td>
<td>Auth/PAA</td>
<td>-</td>
<td>-</td>
<td>AResponse</td>
</tr>
<tr>
<td>13</td>
<td>Auth/PAA</td>
<td>eapSuccess</td>
<td>EAP_SUCCESS</td>
<td>ASuccess</td>
</tr>
<tr>
<td>14</td>
<td>Auth/PAA</td>
<td>eapFail</td>
<td>EAP_FAILURE</td>
<td>AFailure</td>
</tr>
<tr>
<td>15</td>
<td>Auth/PAA</td>
<td>-</td>
<td>EAP_TIMEOUT</td>
<td>ATimeout</td>
</tr>
<tr>
<td>16</td>
<td>Auth/PAA</td>
<td>-</td>
<td>ABORT</td>
<td>AAbort</td>
</tr>
<tr>
<td>17</td>
<td>Peer/PaC</td>
<td>-</td>
<td>DISCARD</td>
<td>CDiscard</td>
</tr>
<tr>
<td>18</td>
<td>Auth/PAA</td>
<td>-</td>
<td>DISCARD</td>
<td>ADiscard</td>
</tr>
</tbody>
</table>
Modelling PANA with CPNs
Motivation

• Verify the PANA specification is accurate and unambiguous

• Gain understanding of PANA's operation

• Gain experience in CPN Tools
Methodology

• Apply steps from a Protocol Engineering Methodology using CPNs and CPN Tools

1. Formal modelling of PANA protocol using CPNs
   • State-based modelling approach using PANA state machines

2. Simulation of the PANA protocol
   • Stepping through selected sequences
   • Generation of Message Sequence Charts

3. State space analysis of PANA
   • Absence of deadlocks, bounds

4. Generation of PANA protocol language
   • To understand interactions between PANA and EAP
PANA Hierarchy and Top Level

VPANA
  VPaC
    VC_INITIAL
      C_INITIAL_Piggyback
      C_INITIAL_No_Piggyback
    C_WAIT_PAA
    C_WAIT_EAP_MSG
    C_WAIT_EAP_RESULT
    C_WAIT_EAP_RESULT_CLOSE
    C_OPEN
    C_WAIT_PNA
    C_SESS_TERM
    C_RETRANSMIT

VPAA
  A_INITIAL
  A_WAIT_EAP_MSG
  A_WAIT_SUCC_PAN
  A_WAIT_FAIL_PAN
  A_OPEN
  A_WAIT_PNA_PING
  A_WAIT_PAN_OR_PAR
  A_SESS_TERM
  A_RETRANSMIT

Communication places
Unordered, reliable

States of PAA
Each page models events/actions of corresponding state
Each state is represented by state table in RFC 5609

CPN Workshop - 19 Oct 2009
PaC

(C_INITIAL, {eap_piggyback=false, RtxTimerOn=false, SessionTimerOn=false, RtxCounter=0, SeqTx=0, SeqRx=INITASEQ})

Client

PaCState

Client2Auth

Auth2Client
PAA

Client2Auth

Auth2Client

Authenticator

(A_INITIALIZ, 
{ eap_piggyback=false, 
RtxTimerOn=False, 
SessionTimerOn=false, 
OptimizedInit=false, 
RtxCounter=0, 
SeqTx=0, 
SeqRx=0, 
Started=false } )
PAA INITIAL state
(A_INITIAL)
Selection of PANA State Table and CPN for PAA in INITIAL state

<table>
<thead>
<tr>
<th>Exit Condition</th>
<th>Exit Action</th>
<th>Exit State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rx:PAN[S] &amp;&amp; ((OPTIMIZED_INIT == Unset)</td>
<td></td>
<td>PAN.exist_avp (&quot;EAP-Payload&quot;))</td>
</tr>
<tr>
<td>Rx:PAN[S] &amp;&amp; (OPTIMIZED_INIT == Set) &amp;&amp; ! PAN.exist_avp (&quot;EAP-Payload&quot;)</td>
<td>None();</td>
<td>WAIT_PAN_OR_PAR</td>
</tr>
</tbody>
</table>

PAN Handling

- RxPAN Start Payload
- RxPAN Start Opt, No PL
- RxPAN Start Not Opt, No PL
- AuthAnswer pan

Client2Auth

Authenticator

msg
Selection of PANA State Table and CPN for PAA in INITIAL state

<table>
<thead>
<tr>
<th>Exit Condition</th>
<th>Exit Action</th>
<th>Exit State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rx:PAN[S] &amp;&amp; ((OPTIMIZED_INIT == Unset)</td>
<td></td>
<td>PAN.exist_avp (&quot;EAP-Payload&quot;)) &amp;if (PAN.exist_avp (&quot;EAP-Payload&quot;)</td>
</tr>
<tr>
<td>TxEAP();</td>
<td></td>
<td></td>
</tr>
<tr>
<td>else {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EAP_Restart();</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SessionTimerReStart (FAILED_SESS_TIMEOUT);</td>
<td></td>
<td></td>
</tr>
<tr>
<td>None();</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAIT_PAN_OR_PAR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Rx:PAN[S] && (OPTIMIZED_INIT == Set) && ! PAN.exist_avp ("EAP-Payload")
input (pan); output (); action (if mscOn then (  
msg. addEvent("Network","PAA",concat [ "AuthAnswer[S,"  
Input.toString (#Seq(pan)),"]");  
msg. addEvent("PAA","EAP Auth","EAP_RESTART"))  
else ()));
Analysis of PANA
State Space Analysis

• Aims
  - Gain understanding of PANA operation
  - Prove the absence of deadlocks in protocol
  - Generate PANA protocol language

• Configurations
  - Piggybacking: Off, On
  - Optimised Initiation: Off, On
  - Maximum retransmit limit at PaC: 0, 1, 2, 3, ...
  - Maximum retransmit limit at PAA: 0
# State Space Analysis: No Retransmissions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Optimised</th>
<th>States</th>
<th>Arcs</th>
<th>Terminals</th>
<th>Client2Auth</th>
<th>Auth2Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Piggyback</td>
<td>Init</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Off</td>
<td>Off</td>
<td>15531</td>
<td>34047</td>
<td>6866</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>On</td>
<td>Off</td>
<td>3436</td>
<td>7212</td>
<td>1265</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Off</td>
<td>On</td>
<td>12079</td>
<td>26360</td>
<td>5292</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>On</td>
<td>On</td>
<td>2085</td>
<td>4233</td>
<td>775</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

- Only Authentication/Authorisation phase considered
- Packet loss is not possible
- Single EAP Request sent by Authenticator
- 4-bit sequence numbers
## Terminal States

- Define expected terminal states based on the protocol state of the PaC/PAA, e.g. Open, Closed, Initial, ...

<table>
<thead>
<tr>
<th>PaC</th>
<th>Initial</th>
<th>WaitPAA</th>
<th>WaitEAP Msg</th>
<th>WaitEAP Result</th>
<th>WaitEAPResultClose</th>
<th>Open</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>PaC Abort</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WaitEAPMsg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WaitSuccPAN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WaitFailPAN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WaitPANPAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open</td>
<td>PaC enters WaitPAA</td>
<td>Success</td>
<td>PaC Abort</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Closed</td>
<td>PAA Abort</td>
<td>PAA Abort</td>
<td>Failed or Abort</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Terminal States

- No unexpected terminal states, but refinement of definition of expected terminal states is necessary

<table>
<thead>
<tr>
<th>PaC</th>
<th>Initial</th>
<th>WaitPAA</th>
<th>WaitEAP Msg</th>
<th>WaitEAP Result</th>
<th>WaitEAPR esultClose</th>
<th>Open</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WaitEAPMsg</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WaitSuccPAN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WaitFailPAN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WaitPANPAR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Open</td>
<td>0</td>
<td>11 7 8 4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>61 41 23</td>
<td>456 61 38</td>
</tr>
<tr>
<td>Closed</td>
<td>0</td>
<td>84 186 68 116</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>164 98 59</td>
<td>6090 871 4660 534</td>
</tr>
</tbody>
</table>

Number of terminal states
Piggyback: Off or On
Optimised Init: Off or On
The Effect of Retransmissions

- Both PaC and PAA may retransmit messages up to a maximum number of times
  - PaC retransmission limit: $M_{R_C}^{PaC}$
  - PAA retransmission limit: $M_{R_C}^{PAA}$
  - Abort when limit is reached

- State space analysis for $M_{R_C}^{PAA} = 0$:
  - Piggyback Off, Optimised Init Off, $M_{R_C}^{PaC} : 0, 1, 2, 3$
  - Piggyback Off, Optimised Init On, $M_{R_C}^{PaC} : 0, 1, 2, 3, 4$
  - Piggyback On, Optimised Init Off, $M_{R_C}^{PaC} : 0, 1, 2, 3, 4, 5, 10$
  - Piggyback On, Optimised Init On, $M_{R_C}^{PaC} : 0, 1, 2, 3, 4, 5, 10$
The Effect of Retransmissions

- Piggybacking on: state space number of states, arcs and terminals expressed as polynomials

\[ S_{MRC_{PaC}} = 441(MRC_{PaC})^2 + 2958(MRC_{PaC}) + 3436 \]
\[ A_{MRC_{PaC}} = 1190.5(MRC_{PaC})^2 + 7029.5(MRC_{PaC}) + 7212 \]
\[ T_{MRC_{PaC}} = 71.5(MRC_{PaC})^2 + 813.5(MRC_{PaC}) + 1265 \]
Piggybacking and Retransmissions

- Retransmissions (by PaC)
  - Piggyback On: PaC can only retransmit ClientInitiation message
  - Piggyback Off: PaC can retransmit ClientInitiation and AuthRequest messages

States identical to states on left, except additional AuthRequest in Client2Auth place
Observations from Analysis

- Terminal markings
  - Valid protocol states for PaC and PAA
  - Closer examination of state variables needed
- Integer bounds on places \textit{Client2Auth} and \textit{Auth2Client}
  - Number of messages sent before waiting for response
  - Bound on \textit{Client2Auth} increases linearly with $MRC_{PaC}$
  - Useful in dimensioning buffer sizes
- State space size
  - Dependent on $MRC_{PaC}$
  - Patterns suggest state space reduction techniques can be applied
Language Analysis

• PANA protocol language: set of sequences of primitives exchanged between PANA and EAP
  - Should match the PANA service language
  - BUT PANA service language not defined!

• Aim: understand the PANA protocol language as step towards defining PANA service language
  - Currently only visual inspection of subsets of the full language
Language Analysis Methodology

ML functions to convert to FSA

State → node number
Binding Element → integer where:
1  -18 if transition models primitive
0  otherwise
Terminal state → halt state

(* Binding Element to Primitive Mapping *)
fun be2str (Bind.C_INITIAL'EAP_RESPONSE (1, _)) = "4"
| be2str (Bind.C_INITIAL'AUTH_USER (1, _)) = "1"
| be2str (Bind.C_INITIAL'RxPAR_Start_No_EAPPayload (1, _)) = "2"
| be2str (Bind.C_WAIT_PAA'RxPAN (1, _)) = "0"
...
## Full PANA Language

### Number of sequences

<table>
<thead>
<tr>
<th>Piggyback</th>
<th>Optimised Init</th>
<th>Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>Off</td>
<td>17862</td>
</tr>
<tr>
<td>Off</td>
<td>On</td>
<td>14742</td>
</tr>
<tr>
<td>On</td>
<td>Off</td>
<td>13604</td>
</tr>
<tr>
<td>On</td>
<td>On</td>
<td>11468</td>
</tr>
</tbody>
</table>
PAA Only Language

Piggybacking On
Optimised Initiation On

CPN Workshop - 19 Oct 2009
Summary

- CPN modelling and analysis of an Internet authentication protocol

- Applied state-based modelling approach to PANA
  - Map state table entries from RFC to CPN transitions

- State space and language analysis
  - Increased confidence in the PANA RFC
  - Refinement of model and analysis techniques necessary
Future Work

- Precise definition of terminal markings
- Analysis of Access, Re-authentication and Termination phases
- Relaxing assumptions (reliable communications, single EAP Request)
- Prove state space results independent of retransmission limits
- Further develop tools for protocol engineering
  - Language analysis, CPNTools+SVN+Diff, generate state tables from model, ...
Thank You

(and the reviewers)