Composable Formal Models for High-Assurance Fault Tolerant Networks

Carolyn Talcott
SRI International

FTN PI meeting January 2003
People

• PIs:
  • Carolyn Talcott (SRI)
  • Jose Meseguer (UIUC)

• Other team members:
  • Steven Eker (SRI)
  • Mark-Oliver Stehr (UIUC)
  • Ambarish Sridharanarayanan (UIUC)
Composable Formal Models for High-Assurance Fault Tolerant Networks

- Sound principles for composition of network reconstitution and recovery services
- Higher assurance for FTN protocols and algorithms
- Compliment red team efforts to harden systems

New Ideas

- Libraries of formal models of attacks
- Formal executable framework for reflective systems
- Formal test bed for design of network reconstitution and recovery services

Impact

- Sound principles for composition of network reconstitution and recovery services
- Higher assurance for FTN protocols and algorithms
- Compliment red team efforts to harden systems

Schedule

<table>
<thead>
<tr>
<th>Task 1: Case Studies</th>
<th>Task 2: Framework Mobile Maude</th>
</tr>
</thead>
<tbody>
<tr>
<td>07/02</td>
<td>07/03</td>
</tr>
</tbody>
</table>

SRI International: Carolyn Talcott
University of Illinois Urbana-Champaign: José Meseguer
Some Possible Case Studies

- Proxy-based Distributed Systems
  - secure distribution of component based systems
- DDOS models
  - formal attack models, formal v&v?
- Secure Spread (Maude + uCAPSL)
  - is it secure?
  - is the group semantics preserved?
- TIARA project
  - intrusion tolerance for ad hoc networks
- Distributed/replicated databases
  - formal verification of core algorithms
  - reuse to verify DB specific optimizations
Analysis of Secure Proxy Toolkit

In collaboration with DC project

Securing Proxy-based Distributed Systems

John Mitchell, Ninghui Li, Derrick Tong
Stanford
Proxy based distributed service
End2end view
Security Goals

• Client VM protected from evil proxy

• Secure communication between client and server proxies

• Client should be able to authenticate service proxy

• Server should be able to authenticate client
The Secure Proxy Toolkit in Pictures
Registering a signed secure proxy

1. RegReq(svcName, Svc)
2. create(sAP)
3. sign(cAPd)
4. register(svcName, signedcAPd)
Getting an authentication proxy

1. findSvc(svcName)
2. lookup(svcName)
3. reply(signedcAPd)
4. verify(signedcAPd, tsK)
5. ok(apxd)
5a. install(apxd)

♦ must check description!
Setting up a secure session

6. authenticate(ccred)
6a. setup secure cnx ---
7. authenticate(ccred)
8. checkClient(ccred, svcName)
9. clientOk(perms)
10. install(sspd)

11. encryptReq(csspd, ccred)
12. encrypted(csspd, ccred)
13. encrypted(csspd, ccred)
14. decrypt(encrypted(csspd, ccred))
15. ok(csspd)
16. install(csspd)
17. findSvcReply(cSP)
16. serviceCall(args)
17. serviceCall(args)
18a. check(args, per)
18. serviceCall(args, cId)

19. serviceReply(result)
20. serviceReply(result)
21. serviceReply(result)

Accessing the service
Proxy Toolkit Models
4 levels of security

- Level 0 -- naive proxy (just does rmi)
- Level 1. naive proxy enhanced with secure communication
- Level 2[t,f]. signed proxy
  - [with, without] checking proxy service name
- Level 3. signed proxy with authentication (and secure session)
Registering a service

Levels 0,1

SvcMgr  sPTKO
spRegister
create
registerReq
registerReply
registerReply

Levels 2,3

SvcMgr  TMS  sPTKO  Lup
spRegister
create
signReq
signReply
registerReq
registerReply
registerReply

Levels differ in choice of proxy behavior
Scenario -- finding and using a service via Level 0,1 PTK

Clear in level 0, secure in level 1

Point of attack
Scenario -- finding and using a service via Level 2 PTK

App  cTMS  cSPTk  sSP  sTMS  Svc  Lup

findService
findServiceReply
check descr?
ok
install
Proxy to wrong service if no check
Illegal Call
Illegal Call

signedProxy
verify
ok
SvcCall
SvcReply
SvcCall
SvcReply
lookUp
Scenario -- finding and using a service via Level 3 PTK
Attack Model I

1. Attacker in the ether + independent imposter
Properties checked:

- 1.1 attacker see/modify client data
  - sent as service arguments or received as reply
- 1.2 attacker replace registered proxy
- 1.3 illegal or unauthorized service call
- 1.4 client imposter succeeds
## Summary of analyses

**attacker in net**

<table>
<thead>
<tr>
<th>Property:</th>
<th>1.1 msg attack</th>
<th>1.2 replace proxy</th>
<th>1.3 illegal call</th>
<th>1.4 imposter succeeds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 0</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Level 1</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Level 2f</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Level 2t</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Level 3</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

yes/no Property holds/not
2. Attacker controls Lookup node
Properties checked:
• 2.1 client app can get proxy to requested/registered service (sanity check)
• 2.2 client app accepts proxy to attacker service
• 2.3 client app accepts wrong proxy
• 2.4 service integrity violated
## Summary of analyses
compromised registry

<table>
<thead>
<tr>
<th>Property:</th>
<th>2.1 sanity check</th>
<th>2.2 bad proxy</th>
<th>2.3 wrong proxy</th>
<th>2.4 impersonate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 0</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Level 1</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Level 2f</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Level 2t</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Level 3</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Yes/no Property holds/not
What Next

- TIARA intrusion tolerance for ad hoc networks
  - abstract router and network model
  - composing - router, TIARA, TCP/IP layer

- DDOS models (JHU/APL)
  - attacks with classification mitigation technology
  - TCP SYN flood attack / Synkill active monitoring

- Secure Spread (Maude + uCAPSL)
  - abstract spread network model
  - compose with Cliques model