Mobile Agent Security

Chris Rygaard, CTO
Aramira™ Corporation
Agenda

- Security requirements
- Trusted third party
  - Multi-jump security
  - Trusted source
  - Itinerary assurance
- Encrypted computing
- Signed code
- Authentication
- Hightened need for traditional security
  (time permitting)
New Security Requirements

- Mobile agents arrive and begin executing
  - Without interaction from user
- With MAs, an attacker might insert unsafe code to be executed, with unsafe data, at any time
- With traditional distributed computing, an attacker can execute known safe code with bad data, perhaps at a bad time
Security Needs

- Exchange mobile agents with untrusted hosts
  - Framework must not facilitate an attack
- Ensure unsafe code does not attack
  - Even if mobile agent has passed through a hostile host
- Also requires traditional distributed security
Security Needs (continued)

- Protect MA from hostile host
  - Termination
    - Might not be desirable
  - Modification
  - Correct execution

- Protect host from hostile MA
  - Resources
  - Data
  - User
Jumping Beans: Trusted Third Party

- Star architecture enables Jumping Beans' security
- Hosts never communicate directly with each other, hosts communicate with MaSC only
- MAs pass through MaSC on each jump
- Allows secure exchange of MAs with untrusted hosts

- Reinforce MA on each jump
- Management point
Jumping Beans: Trusted Third Party

4 independent layers of security
1. Traditional distributed security
2. Trusted source
3. Multi-jump security
4. Monitoring and intervention

If one layer fails, the system is still protected by remaining three layers
Multi-Jump Security

- A host needs to protect itself from MAs
  - Can be implemented with Java's interceptors
- But how much should a host trust an MA?
Multi-Jump Security (continued)

- Each MA is trusted no more than the least trusted host it has visited
  - Each host is assigned a level of trust
    - Fine-grained
    - Looks like an ACL
  - MA's ACL is a most-restrictive composite
    - From ACLs of all previously visited hosts
Multi-Jump Security (continued)

Mobile app after jump 1:
Can't open file "x"
Can't open any socket
etc.

Host #1
Can't open file "x"

Host #2
Can't open socket 10.14.16.81

Host #3
Can't delete file "y"
Can't edit own itinerary

Mobile app after jump 2:
Can't open file "x"
Can't open socket 10.14.16.81

Host #4
Can't delete any file

Mobile app after jump 3:
Can't open file "x"
Can't open socket 10.14.16.81
Can't delete file "y"
Can't edit own itinerary

Host #5
Can open only file "z"
Can't open any socket

Mobile app after jump 4:
Can't open file "x"
Can't open socket 10.14.16.81
Can't delete any file
Can't edit own itinerary

Host #6
Can't open file "x"
Trusted Third Party: Multi-Jump ACL Assurance

Inspect ACL on each jump
Trusted Source

- Accept code from trusted hosts only
  - System administrator designates which hosts are trusted to transmit code to other hosts
    - At MaSC
- Search protocols locate needed resources from hosts which are designated as trusted
- Partial protection of MA from hostile host
Trusted Source (continued)
Trusted Source (continued)

Host

Jumping Beans
Management and
Security Console

Host

Host

Host

Host

Enter Data:

(trusted)

(untrusted)

(trusted)

(untrusted)
Trusted Source (continued)

For $i = 0$ to $9$

doIt();
while (go)

copy (trusted)

copy (trusted)

copy (trusted)

copy (trusted)

Jumping Beans Management and Security Console

doi();
while (go)

copy (trusted)

copy (trusted)

copy (trusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (untrusted)

untrusted (un trusted)
Trusted Source (continued)

```
for (i=0; i<10)
doit();
while (go)
saved
```

Jumping Beans Management and Security Console

Enter Data:

(trusted)

(trusted)

(untrusted)

(untrusted)

Host

Host

Host

Host

Host

Host

Host

Host
Trusted Source *(continued)*

```c
for (i=0; i<10) doIt();
while (go)
    for (i=0; i<10) doIt();
    while (go)
```

Jumping Beans Management and Security Console
- saved
- new
- compare
Trusted Source (continued)

For (i=0; i<10) doIt();
while (go);

Jumping Beans Management and Security Console

Enter Data:

(trusted)
Host

(untrusted)
Host

(trusted)
Host

(untrusted)
Host
Trusted Source (continued)

(trusted)
Host

(untrusted)
Host

Jumping Beans Management and Security Console

for (i=0; i<10) doIt();
while (go)
Overwrite

(untrusted)
Host

(trusted)
Host

(trusted)
Host

(untrusted)
Host
Trusted Source (continued)

```
for (i=0; i<10)
doIt();
while (go)
saved
```

Enter Data:

```
20
```
Trusted Source (continued)

(trusted) Host

(untrusted) Host

Jumping Beans Management and Security Console

for (i=0; i<10)
doit();
while (go)

for (i=0; i<10)
doit();
while (go)

(saved) new

compare

Host

Host

Host

Host

(trusted)

(untrusted)
Trusted Source (continued)

Host

Jumping Beans Management and Security Console

for (i=0; i<10) doIt();
while (go) saved

Host

Host

Host

(trusted)
(untrusted)
(trusted)
(untrusted)
Trusted Source (continued)
Itinerary Assurance

- Where has it been?
- Where will it go?
- Will it actually go there?
Trusted Third Party: Itinerary Assurance
Trusted Third Party

Strengths
- Ideal for IT shop in large corporation
- The MaSC is the TCB
  - In peer-to-peer, every host must be part of TCB

Limitations
- System administrator is trusted by all
  - Not good for mass-market
- MaSC is single point of security failure
- MaSC is bottleneck
Encrypted Computing
(Research by Christian Tschudin)

1. Determine required computation
2. Obfuscate data and/or computations
3. Send to other computer
4. Other computer performs operations
   a. Other computer cannot discover what it has computed
5. Send results home
6. De-obfuscate results
J wants K to compute $x \times y$ without discovering $x$, $y$, or the product.

1. J generates random numbers $R_1$ and $R_2$
2. J computes $P_1 = R_1 \times x$
3. J computes $P_2 = R_2 \times y$
4. J sends $P_1$ and $P_2$ to K
5. K computes $F = P_1 \times P_2$
6. K returns $F$ to J
7. J computes result = $F / (R_1 \times R_2)$
Encrypted Computing: Limitations

- Not yet a general programming language
  - Limited to computing polynomials
- Even if it were expanded to general programming language:
  - Can't hide file access, user interaction, I/O, etc.
  - It is very CPU intensive for both computers
Signed Code

- Executable file signed by author
  - Prevents tampering
- Limitations
  - Proving authorship does not determine level of trust
  - Certificate management?
  - Does not address other security issues
- Strength
  - Conceptually simple
Authentication

▲ Important technique to protect compromised hosts
▲ Jumping Beans is intelligent about when to issue a challenge
  ● Protects stolen device, even before theft is discovered
Conclusions

- MAs can be exchanged with untrusted hosts
  - *Academia, DOD have not discovered this!*

- Problems yet to be solved
  - Correct execution of MA by hostile host
    - Partial solutions available
  - Termination of MA by hostile host
    - Probably shouldn't be solved
  - MAs for mass market
Research: Applications of Mobile Agents

- Intrusion response
  - "White Blood Cells" to eradicate intruders
- Management of heterogeneous systems
  - Send management software on the fly
Mobile Agent Security

Heightened Need for Traditional Security
Heightened Requirements for Traditional Security

- Mobile applications are extremely powerful
- MAs must be properly handled
  - And they will solve many problems
  - Will cause problems if mishandled
Signatures Must Keep Hackers Out

- With traditional distributed computing, an attacker can execute known safe code with bad data, perhaps at a bad time.
- With MAs, an attacker might insert unsafe code to be executed, with unsafe data, at any time.
- Signatures provided by Jumping Beans.
Monitoring and Intervention are Critical

If something is amiss, system administrator must have tools to diagnose and isolate any problems
- Lost or stolen device
- Rogue MA

Jumping Beans' provides this
- Quarantine or sever any branch of star
- Real-time display of activity at MaSC
- Destroy, dispatch, deactivate, etc. any MA
  - From MaSC
Audit Logs

- Attacker is less likely to strike if he/she can be discovered
- Jumping Beans has indelible audit logs

- Logs can't be attacked by rogue MA
- MAs don't grow too big
Authentication

⚠️ Critical for mobile devices
- If a mobile device falls into wrong hands, it must not be a door into computing system

⚠️ Jumping Beans is intelligent about when to issue an authentication challenge
- Protects lost or stolen device
  - Even if loss or theft is not yet discovered
- Currently password based
Nonrepudiation

Because of MAs, interaction between hosts can be very *ad-hoc*

Evidence must be kept for settling any sort of dispute

Provided in Jumping Beans

**Diagram:**
- **Computer J**
  - J can prove:
    - Y was sent
    - Y came from K
    - Y was intended for J
    - Y is in response to X, not some other message

- **Computer K**
  - K can prove:
    - X was sent
    - X came from J
    - X was intended for K

- Message X
- Response Y
Encryption

- Provides privacy, just like other systems
- Probably no more important for MAs than for other distributed computing systems
- Provided by Jumping Beans
Replay Prevention

- Difficult to assess importance to MAs, as compared to other computing systems
  - But it is very important, just like other computing systems
- Provided by Jumping Beans
Manageability

⚠️ If the security is difficult to manage, users, developers, and system administrators will turn it off

- Jumping Beans is easily managed from Management and Security Console
  - Inheriting groups: easily manage many groups
Jumping Beans' Security Management

- Jumping Beans' security is transparent to system administrator
  - Just configuration
- Jumping Beans security is transparent to app developer
  - Although app developer can interact through API
- Jumping Beans security is transparent to end users