Security Analysis of Network Protocols

John Mitchell      Vitaly Shmatikov
Stanford          SRI

http://www.stanford.edu/class/cs259/

Course organization

- Lectures
  - Tues, Thurs for approx first six weeks of quarter
  - Project presentations last three weeks
- This is a project course
  - There may be one or two short homeworks
  - Most of your work will be project and presentation

Please enroll!

Computer Security

- Cryptography
  - Encryption, signatures, cryptographic hash, ...
- Security mechanisms
  - Access control policy
  - Network protocols
- Implementation
  - Cryptographic library
  - Code implementing mechanisms
    - Reference monitor and TCB
    - Protocol
  - Runs under OS, uses program library, network protocol stack

Analyze protocols, assuming crypto, implementation, OS correct

Cryptographic Protocols

- Two or more parties
- Communication over insecure network
- Cryptography used to achieve goal
  - Exchange secret keys
  - Verify identity (authentication)

Class poll:
  - Public-key encryption, symmetric-key encryption, CBC, hash, signature, key generation, random-number generators

Correctness vs Security

- Program or System Correctness
  - Program satisfies specification
    - For reasonable input, get reasonable output
- Program or System Security
  - Program properties preserved in face of attack
    - For unreasonable input, output not completely disastrous
- Main differences
  - Active interference from adversary
  - Refinement techniques may fail

Security Analysis

- Model system
- Model adversary
- Identify security properties
- See if properties preserved under attack

Result
  - No "absolute security"
  - Security means: under given assumptions about system, no attack of a certain form will destroy specified properties.
Important Modeling Decisions

◆ How powerful is the adversary?
  ◆ Simple replay of previous messages
  ◆ Block messages; Decompose, reassemble and resend
  ◆ Statistical analysis, partial info from network traffic
  ◆ Timing attacks
◆ How much detail in underlying data types?
  ◆ Plaintext, ciphertext and keys
    – atomic data or bit sequences
  ◆ Encryption and hash functions
    – “perfect” cryptography
    – algebraic properties: \( \text{encr}(x \cdot y) = \text{encr}(x) \cdot \text{encr}(y) \) for
      \( \text{RSA}\text{-}\text{encr}(k, \text{msg}) = \text{msg}^k \mod N \)

This has been our research area

◆ Automated nondeterministic finite-state analysis
  • General paper, Oakland conference, 1997 [JM, …]
  • Efficiency for large state spaces, 1998 [VS, …]
  • Analysis of SSL, 1998-99 [VS, JM, …]
  • Analysis of fair exchange protocols, 2000 [VS, JM, …]
◆ Automated probabilistic analysis
  • Analysis of probabilistic contract signing, 2004 [VS, …]
  • Analysis of an anonymity system, 2004 [VS, …]
◆ Beyond finite-state analysis
  • Decision procedures for unbounded # of runs
  • Proof methods, assuming idealized cryptography
  • Beyond idealized cryptography

Many others have worked on these topics too …

Some other projects and tools

◆ Exhaustive finite-state analysis
  ◆ FDR, based on CSP [Lowe, Roscoe, Schneider, …]
◆ Search using symbolic representation of states
  ◆ Meadows: NRL Analyzer, Millen: Interrogator
◆ Prove protocol correct
  ◆ Paulson’s “Inductive method”, others in HOL, PVS, …
  ◆ MITRE -- Strand spaces
  ◆ Process calculus approach: Abadi-Gordon spi-calculus, applied pi-calculus, …
  ◆ Type-checking method: Gordon and Jeffreys, …

Many more – this is just a small sample

Example: Needham-Schroeder

◆ Famous simple example
  • Protocol published and known for 10 years
  • Gavin Lowe discovered unintended property while preparing formal analysis using FDR system
◆ Background: Public-key cryptography
  • Every agent A has
    – Public encryption key \( K_a \)
    – Private decryption key \( K_a^{-1} \)
  • Main properties
    – Everyone can encrypt message to A
    – Only A can decrypt these messages

Anomaly in Needham-Schroeder

Evil agent E tricks honest A into revealing private key NB from B
Evil E can then fool B

Needham-Schroeder Key Exchange

\[ \{ A, \text{Nonce}_A \} K_B \]
\[ \{ \text{Nonce}_A, \text{Nonce}_B \} K_A \]
\[ \{ \text{Nonce}_B \} K_B \]

Result: A and B share two private numbers not known to any observer without \( K_a^{-1}, K_b^{-1} \)
Explicit Intruder Method

Mur$\phi$ [Dill et al.]

Finite-state methods

Verification vs Error Detection

Applying Mur$\phi$ to security protocols

Needham-Schroeder in Mur$\phi$ (1)
Needham-Schroeder in Murϕ (2)

```plaintext
MessageType : enum {
  M_NonceAddress, -- {Na, A}Kb nonce and addr
  M_NonceNonce, -- {Na,Nb}Ka two nonces
  M_Nonce -- {Nb}Kb one nonce
};
Message : record
  source: AgentId; -- source of message
  dest: AgentId; -- intended destination of msg
  key: AgentId; -- key used for encryption
  mType: MessageType; -- type of message
  nonce1: AgentId; -- nonce1
  nonce2: AgentId; -- nonce2 OR sender id OR empty
};
```

```
--- intruder i sends recorded message
ruleset i: IntruderId do -- arbitrary choice of
  choose j: int[i].messages do -- recorded message
  ruleset k: AgentId do -- destination
    rule "intruder sends recorded message"
    !ismember(k, IntruderId) & -- not to intruders
    multisetcount (l:net, true) < NetworkSize
  ==>
  var outM: Message;
  begin
  outM := int[i].messages[j];
  outM.source := i;
  outM.dest := k;
  multisetadd (outM, net);
  end; end; end; end;
```

Adversary Model

- **Formalize "knowledge"**
  - initial data
  - observed message fields
  - results of simple computations
- **Optimization**
  - only generate messages that others read
  - time-consuming to hand simplify
- **Possibility: automatic generation**

Run of Needham-Schroeder

- **Find error after 1.7 seconds exploration**
- **Output: trace leading to error state**
- **Murϕ times after correcting error:**

<table>
<thead>
<tr>
<th>number of init. res. int.</th>
<th>size of network</th>
<th>states</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1706</td>
<td>3.1s</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>40207</td>
<td>82.2s</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>17277</td>
<td>43.1s</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>514550</td>
<td>5761.1s</td>
</tr>
</tbody>
</table>

Limitations

- **System size with current methods**
  - 2-6 participants
  - Kerberos: 2 clients, 2 servers, 1 KDC, 1 TGS
  - 3-6 steps in protocol
  - May need to optimize adversary
- **Adversary model**
  - Cannot model randomized attack
  - Do not model adversary running time
Security Protocols in Murϕ

- Standard "benchmark" protocols
  - Needham-Schroeder, TMN, ...
  - Kerberos

- Study of Secure Sockets Layer (SSL)
  - Versions 2.0 and 3.0 of handshake protocol
  - Include protocol resumption

- Tool optimization

- Additional protocols
  - Contract-signing
  - Wireless networking

... ADD YOUR PROJECT HERE ...

Plan for this course

- Protocols
  - Authentication, key establishment, assembling protocols together (TLS ?), fairness exchange, ...

- Tools
  - Finite-state and probabilistic model checking, constraint-solving, process calculus, temporal logic, proof systems, game theory, polynomial time ...

- Projects
  - Choose a protocol or other security mechanism
  - Choose a tool or method and carry out analysis
  - Hard part: formulating security requirements

Reference Material (CS259 web site)

- Protocols
  - Clarke-Jacob survey
  - Use Google; learn to read an RFC

- Tools
  - Murphi
    - Finite-state tool developed by David Dill's group at Stanford
  - PRISM
    - Probabilistic model checker, University of Birmingham
  - MOCHA
    - Alur and Henzinger; now consortium
  - Constraint solver using prolog
    - Shmatikov and Milner
  - Isabelle
    - Theorem prover developed by Larry Paulson in Cambridge, UK
    - A number of case studies available online

Hope you enjoy the course

- We'll lecture for a few weeks to get started
  - Case studies are the best way to learn this topic
  - Cathy Meadows guest lecture next Thursday

- Choose a project that interests you !!!
  - If you have another idea, come talk with us
  - Can build or extend a tool, or paper study if you prefer

Protocols and other mechanisms

- Secure electronic transactions (SET) or other e-commerce protocols
- Onion routing or other privacy mechanism
- Firewall policies
- Electronic voting protocols
- Publius: censorship-resistant Web publishing
- Group key distribution protocols
- Census protocols
- Stream signing protocols:
  - Analysis/verification/defense against MCI's network routing scam
    - Apparently, MCI routed long-distance phone calls through small local companies and Canada to avoid paying access charges to local carriers
- Wireless networking protocols