
Orna Grumberg
Technion, Israel

Joint work with Adi Sosnovich and Gabi Nakibly

Tel-Aviv University, December 5, 2013
Motivation

• Attacks on network protocols, taking advantage of built-in vulnerabilities, are not easy to identify

- Rely on legitimate functionality of the protocol
- May involve only a small number of messages
- Identifying attacks is done mostly manually, by experts, in an ad hoc manner
Goals

• Develop **automatic** methods for identifying **attacks** in network protocols

• Using methods and tools for **formal verification** of software and hardware
  - Model checking
Model Checking \([CE81, QS82]\)

An efficient procedure that receives:

- A finite-state model describing a system
- A temporal logic formula describing a property

It returns

yes, if the system has the property
no + Counterexample, otherwise
Mutual Exclusion Example

- Two process mutual exclusion with shared semaphore
- Each process has three states
  - Non-critical (N)
  - Trying (T)
  - Critical (C)
- Semaphore can be available (\(\text{sem}=1\)) or taken (\(\text{sem}=0\))
- Initially both processes are in the Non-critical state and the semaphore is available --- \(N_1 N_2 S_0\)

- \(S_0\) denotes \(\text{sem}=0\)
- \(S_1\) denotes \(\text{sem}=1\)
Mutual Exclusion Example

\[ P = P_1 \parallel P_2 \]

\[ P_i :: \text{while (true) } \{ \]
\[ \quad \text{if (} v_i \text{ == N) } v_i = T; \]
\[ \quad \text{else if (} v_i \text{ == T && sem=1) } \]
\[ \quad \quad \{ v_i = C; \text{ sem=0;} \} \]
\[ \quad \text{else if (} v_i \text{ == C) } \{ v_i = N; \text{ sem=1;} \} \]
\[ \} \]

Initial state: \((v_1 == N, v_2 == N, \text{ sem=0})\)
Mutual Exclusion Example

\[ M \models AG \rightarrow (C_1 \land C_2) \]

*The two processes are never in their critical states at the same time*

The state with \((C_1 \land C_2)\) is not reachable
Mutual Exclusion Example

\[ M \models AG \rightarrow (C_1 \land C_2) \]
Mutual Exclusion Example

\[ M \models AG \rightarrow (C_1 \land C_2) \]
Mutual Exclusion Example

\[ M \models AG \not\rightarrow (C_1 \land C_2) \]

\[ S_2 \]
Mutual Exclusion Example

\[ M \models AG \rightarrow (C_1 \land C_2) \]
Mutual Exclusion Example

\[ M \models AG \rightarrow (C_1 \land C_2) \]

\[ S_4 \subseteq S_0 \cup \ldots \cup S_3 \]
Mutual Exclusion Example

The two processes are never in their trying states at the same time

M ⊨ AG \neg (T_1 \land T_2)
Mutual Exclusion Example

\[ M \models AG \neg (T_1 \land T_2) \]
Mutual Exclusion Example

\[ M \models AG \neg (T_1 \land T_2) \]
Mutual Exclusion Example

\[ M \not\models AG \rightarrow (T_1 \land T_2) \]

A violating state has been found
Mutual Exclusion Example

M ≠ AG → (T₁ ∧ T₂)

Model checking returns a counterexample
Our goals

To search for attacks using model checking

For this purpose, we define:

• **Model**
  - Represents the protocol’s behaviors
  - Includes an attacker with predefined capabilities

• **Specification**
  - Specifies “suspect” states
Challenges

- Building a model which is
  - Sufficiently detailed: to enable identifying attacks based on the protocol's functionality
  - Sufficiently reduced: feasible for model checking tools

- Write general specification to identify different kinds of attacks with different techniques
Advantages of our approach

• We do not need to define an attack, but only its possible outcome.

  - Specifying suspect states requires less knowledge and efforts than defining an attack

  - May enable finding new attacks, unknown by now
Routing in the Internet

• How do packets get from A to B in the Internet?
Routing in the Internet

- Each router makes a local decision on how to forward a packet towards B.
Research Focus - OSPF

- We focused on the routing protocol Open Shortest Path First (OSPF)

- OSPF is widely used for routing in the Internet
  - Finding attacks on OSPF is significant

- OSPF is a complex protocol
  - We may be able to derive insights from its modeling to modeling of other network protocols
OSPF

- Each router compiles a database of the most recent OSPF messages received from all routers in the network.

Using this database a router obtains a complete view of the network topology.
OSPF

- OSPF messages are **flooded** through the network
OSPF Attacks

- The goal of an OSPF attacker is to advertise fake messages on behalf of some other router(s) in the network.

<table>
<thead>
<tr>
<th>Originator</th>
<th>List of neighbors</th>
<th>Links costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>r5</td>
<td>r3,r8</td>
<td>...</td>
</tr>
</tbody>
</table>
OSPF Attacks

Routing path before from A to B

Routing path after from A to B
When a router receives a message in its own name that it didn't originate, it sends a **fight back** message to all its neighbors.

The fight back message is supposed to revert the effect of the attack eventually.
OSPF Attacks

• An attack is a run of the protocol that creates a fake topology view for some routers in the network

• An attack is called persistent if the fake topology view remains in some routers' databases

• We are interested in finding persistent attacks
OSPF Concrete Model

- A **fixed** network topology
- **Router Model**
  - Models a legitimate router
- **Attacker Model**
  - Models a malicious router
    - can send any *random message* to any *random destination* router
    - can *ignore* incoming messages.
In our model:

• **Messages originated by the attacker are marked with a special flag** `isFake`

• **This flag is not part of the OSPF standard, and legitimate routers do not make use of it**

• **This flag allows us to easily define the specifications for the model**
OSPF Concrete Model

• Our formal model for OSPF is a finite state machine with global states and transitions

• The model is a simplified version of OSPF, which includes the fight back mechanism
Specification

• A global state is considered **attacked** if:
  - Some router has a fake message in its database
  - No message resides in any router's queue

• An attacked state defines the outcome of a successful persistent attack regardless of a specific attack technique
Model Checking

• We implemented the model of OSPF in C, and used the Bounded Model Checking tool CBMC to find persistent attacks on OSPF

• A counterexample returned by CBMC is an attack
Example of Attacks on OSPF

Attack #1

- The attacker (r3) originates a fake message:
  dest = r2, orig = r4
Example of Attacks on OSPF

Attack #2

- The attacker (r3) sends two fake messages:
  - $m_1 = (\text{dest} = r4, \text{orig} = r1, \text{sequence\_number} = 1)$
  - $m_2 = (\text{dest} = r4, \text{orig} = r1, \text{sequence\_number} = 2)$
Another demonstration of attack #2 on a different topology
Concrete Model - Problems

- state explosion problem

- Models that can be handled are very small in size and hence restricted in their topologies and functionality

- We would like to extend our search for attacks to larger and more complex topologies
Abstract Model

• We are interested in general attacks
  - insensitive to most of the topology's details
  - can be applied in a family of topologies

• We define an abstract model which:
  - represents a family of concrete models
  - under-approximates each member in the family.
Abstract Model

• The abstract model consists of an abstract topology and an abstract protocol

• We defined several levels of abstract components

• An abstract topology may also contain some unabstracted routers

• The attacker is always an un-abstracted router
Main Property of the Abstract Model

• If an attack is found on an abstract network, then there is a corresponding attack on each one of the concrete networks represented by it.
Example of an Abstract Attack on OSPF in the Abstract Model

- The attacker sends a fake message with:
  dest=2, orig=4
Example of an attack in a concrete instantiation of the abstract model
Example of a similar attack on another possible instantiation of the abstract model.
Examples of attacks on OSPF in the abstract model

• Attack # 2

- The attacker (designated router) originates a fake message on behalf of sr1:
  \[ m = (\text{dest} = \text{sr}5; \text{orig} = \text{sr}1; \text{seq} = 1; \text{isFake} = T) \]
Correctness of Our Method

• Lemma

- For each abstract transition on the abstract topology, there is a corresponding concrete finite run on each matching concrete topology
Correctness of Our Method

• Theorem
  - An abstract attack found on an abstract topology $T_A$, has a corresponding attack on each matching concrete topology $T_C$. 
Exposed OSPF vulnerabilities:

- a message is opened only by its destination
- the flooding procedure does not flood a message back to its source

- As a result, a fake message in the name of router r might be sent through r

- If the attacker plays the role of a designated router, then by ignoring messages it can stop message flooding, including fight back messages
Conclusion

• We automatically found attacks on small concrete models

• We automatically found general attacks on small abstract models

• The general attacks are applicable to huge networks, with possibly thousands of routers
  - No model checker can be applied directly to such networks
Conclusion

• We developed a novel technique for parameterized networks suitable for finding a counterexample (in our case an attack) on each member of the family
Thank You