
Adi Sosnovich, Orna Grumberg, and Gabi Nakibly

CAV’13
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 model checking
In order to search for attacks using model checking

We need to define:

- **Model**
  - Representing the protocol’s behaviors
  - Including an *attacker* with predefined capabilities

- **Specification**
  - Specifying “*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

<table>
<thead>
<tr>
<th>Originator</th>
<th>List of neighbors</th>
<th>Links costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>r6</td>
<td>r4, r1, r8</td>
<td>...</td>
</tr>
</tbody>
</table>
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
  - A global state contains databases and message queues of all routers

- 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:
  
  \[
  \begin{align*}
  m1 &= (\text{dest} = r4, \text{orig} = r1, \text{sequence_number} = 1) \\
  m2 &= (\text{dest} = r4, \text{orig} = r1, \text{sequence_number} = 2)
  \end{align*}
  \]
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 un-abstracted 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 # 3

- The attacker (designated router) originates a fake message on behalf of sr1:
  \( m = (\text{dest} = \text{sr}5; \text{orig} = \text{sr}1; \text{seq} = 1; \text{isFake} = \text{T}) \)
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
  - Built in security vulnerabilities

• The general attacks are applicable to huge networks, with possibly thousands of routers
  - No model checker can be applied directly to such networks
• Recently, Nurit Devir, as part of her master, exploited machine learning to find attacks on OSPF at runtime
Summary:
We learned in the course:

- Floyd method for partial and full correctness
- Temporal logics
- Explicit CTL model checking
- BDD-based CTL model checking
- SAT-based bounded model checking
• Equivalences and preorders over Kripke structures
• Abstraction and refinement
We did not learn in the course:

- SAT-based full verification
- LTL model checking
- Automata-theoretic approach to model checking
- Compositional verification using automata learning
• Applications:
  - Security vulnerabilities
  - Repair
  - Differential analysis
  - ......

Will be learned in the advanced course

• Synthesis
• Probabilistic model checking
• Real-time and hybrid systems
Thank You