Mutation Testing of Protocol Messages Based on Extended TTCN-3

Chuanming Jing $^{1,2}$, Zhiliang Wang$^{1,3}$, Xingang Shi$^{1,3}$, Xia Yin$^{1,2}$, Jianping Wu$^{1,2,3}$

1 Tsinghua National Laboratory for Information Science and Technology
2 Department of Computer Science & Technology, Tsinghua University
3 Network Research Center of Tsinghua University
Outline

- Motivation and related work
  - Formal model
  - Test case generation
  - Extension of TTCN-3
  - Test practice
- Conclusions
Outline

- Motivation and related work
  - Formal model
  - Test case generation
  - Extension of TTCN-3
  - Test practice
  - Conclusions
Summary: Cisco IOS 12.0 through 12.3YL, with BGP enabled and running the bgp log-neighbor-changes command, allows remote attackers to cause a denial of service (device reload) via a malformed BGP packet.

Published: 5/2/2005

CVSS Severity: 5.0 (Medium)
Why robustness testing

- Conformance testing has limitation
  - partially specified
  - optional requirements
- Scale and complexity of Internet surges
  - noise, disturbance, misconfiguration and vicious man-made attacks
What is robustness testing

- the test in the presence of invalid inputs or stressful environmental conditions (IEEE STD 610.12)
  - detect vulnerabilities
    - Input validation
    - malformed message parsing
    - state transitions
  - overlap but different from security testing
- mutation testing of messages
  - Data-driven robustness testing
Related works (1)

- Model-based robustness testing
  - Difficult to guide test practice

- Fuzz Testing
  - Deliver semi-valid data to the target
  - Widely used in software testing
  - manual, not efficient, not genetic
Related works (2)

- Deficiencies
  - lacks guidance of theory
  - verdict mechanism
  - structure of test case
  - the readability, maintainability of test suite
Outline

- Motivation and related work
  - Formal model
- Test case generation
- Extension of TTCN-3
- Test practice
- Conclusions
Our model

- **NFSM** $M=\langle S, s_0, I, O, T \rangle$
  - $I = I_{\text{spec}} \cup I_{\text{unspec}}$
  - $S = S_{\text{spec}}$
    - $S_{\text{spec}} = \{s_i \mid i=1,2,\ldots\}$, $s_i \in S_i \subseteq S_{\text{spec}}$
    - $T = T_{\text{deter}} \cup T_{\text{nondeter}}$

- **Benefits**
  - inputs and transitions
  - for complex protocol, EFSM is better.

- **Anomalous Test Case** $= \langle \text{State Leading Sequence, Invalid PDU Inputting, Normal-Verification Sequence} \rangle$
How to construct Normal-Verification Sequence

- After faulty injection, $t \in T_{\text{deter}}$
  - State verification sequence of $s_j$
    - UIO sequence
- After faulty injection, $t \in T_{\text{nondeter}}$
  - State identification sequence of $s_{?j}$
    - Distinguish sequence
  - As the difficulty of State identification, in test practice, we use another sequence
    - Forced State Transition
Forced state transition

$S_i \xrightarrow{t_j \in T_{\text{nondeter}}} S_{k1}$

$S_{k1} \xrightarrow{i_j \in I} S_i$

$S_{k2} \xrightarrow{t_j \in T_{\text{nondeter}}} S_{k3}$

$S_{k3} \xrightarrow{i_j \in I} S_i$

$S_i \xrightarrow{t_j \in T_{\text{nondeter}}} S_{?k}$

Forced Transition

$S_{?k} \xrightarrow{i_j \in I} S_i$

$S_i \xrightarrow{t_j \in T_{\text{nondeter}}} S_j$

$S_j \xrightarrow{t_j \in T_{\text{nondeter}}} S_i$
Motivation and related work

Formal model

Test case generation

Extension of TTCN-3

Test practice

Conclusions
Invalid Inputs & test case

Invalid Inputs Generation

- Field value mutation rules
  - Boundary value
  - Input partition value
- Field mutation rules
  - Removal and Addition
  - Overflow
  - Permutation

Why compound anomalous test case

- Test requirement (stressful condition)
- Simplify the sequence
Outline

- Motivation and related work
  - Formal model
  - Test case generation
  - Extension of TTCN-3
- Test practice
- Conclusions
Why we use TTCN-3

- TTCN-3
  - By ETSI. As a standard testing language, has many advantages and Widely used

- Extension
  - Not good for mutation operation
  - Difficult for Test case description
testcase  Onebyone_HL1_Opt( )

  runs on MyTestComponentAsync

  system  SystemComponent   {
    map(mtc:MyPortAsync, system:SystemPort1);
    P1( );
  }

  loopreplace HL1.Opt Boundary8bit_value   {
    Onebyone_HL1( );
  }

  Normal_Verification( );

  stop;
}
Outline

- Motivation and related work
  - Formal model
  - Test case generation
  - Extension of TTCN-3
- Test practice
- Conclusions
Test practice: OSPFv2

- Forced transition
  - “1-Way”, “KillNbr”, “InactivityTimer”, “SeqNumberMismatch” and “BadLSReq”

- Zebra
  - cannot parse invalid messages with mutated “length” field in OSPF header with robustness
Outline

- Motivation and related work
  - Formal model
  - Test case generation
  - Extension of TTCN-3
  - Test practice
- Conclusions
Conclusions & Future Work

- Contribution
  - a formal model
  - test verdict
  - test data set
  - compound test cases
  - apply TTCN-3

- Future Work
  - focus on application layer protocols
  - apply our method to distributed systems
Thank you!
OSPF State Machine

1 -- Hello Received
2 -- Hello Sent out
3 -- DD packet Received
4 -- DD packet sent out
5 -- LSR packet received
6 -- LSR packet sent out
7 -- LSU packet received
8 -- LSU packet sent out