Hypothesis Testing for Network Security

Philip Godfrey, Matthew Caesar, David Nicol,
William H. Sanders, Dong Jin
INFORMATION TRUST INSTITUTE
University of Illinois at Urbana-Champaign
We need a science of security

• Practice of doing cyber-security research needs to change
  – Attempts based on reaction to known/imagined threats
  – Too often applied in ad-hoc fashion

• SoS program: move security research beyond ad-hoc reactions
  – Need a principled and rigorous framework
  – Need a *scientific* approach
What is science?

**science**  *noun* \ˈsī-ən(t)s\  
: the **systematic study** of the structure and behavior of the natural and physical world through observation and experiment

The scientific method

1. Ask a question
2. Formulate a hypothesis
3. Design and conduct an experiment
4. Analyze results
Towards a science of security

• Can we apply the scientific method to the domain of cybersecurity?
  – Challenges: complex, large scale+dynamic environments, many protocols/mechanisms
  – Opportunities: isolation, rigorous analyses, formal models, automation

• Can we develop a methodology for science of security?
Our work

• NetHTM: a methodology for science of security
  – Techniques for performing/integrating security analyses to rigorously answer hypotheses about end to end security of a network

• Core: hypothesis evaluation engine
  – Input: testable hypotheses, formal model of system
  – Automatically designs and conducts experiments to evaluate veracity of hypotheses

• Our focus: Network data flow security
  – Builds upon our prior work in formal network modeling
Overall System Architecture

Hypotheses

- “All network paths traverse a firewall”
- “Fraction of CRE vulnerabilities in network, given set of deployed ACLs, is less than 1%”

Security Scientist

NetHTM Hypothesis Testing Platform

System under evaluation

Results
Active sub-tasks and Status

• Task 1: Methodologies for modeling and analyzing networks
  - Core Network Model
  - Modeling virtualized networks [best paper award, HotSDN 2014]

• Task 2: Automated techniques for hypothesis testing
  - Automated experiment construction algorithm
  - Database model of network behavior

• Task 3: Realizing a practical system
  - Modeling dynamic behaviors [NSDI 2015]
Let’s start with a router

- Configuration
- Control Plane
- Data Plane
- Network Forwarding
One approach: Build a model of the router

Pros:
- Can test prior to deployment

Cons:
- Modeling is complex
- Prediction misses bugs in control plane
- Requires vendor support
Our approach: Just model the data plane

Pros:
- Checks as close as possible to network behavior
- Unified analysis for multiple protocols
- Catches implementation bugs
Our approach: Data-plane modeling

• **Challenge:** need some general way to express complex forwarding behavior

• **Solution:** Represent data plane as boolean functions
  – Can leverage well-understood approaches to SAT solving, to check hypotheses against data plane
  – Translate SAT results to report hypothesis veracity along with diagnostic information
Examples

Packet Filtering

<table>
<thead>
<tr>
<th>Destination</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.1.0/24</td>
<td>v</td>
</tr>
</tbody>
</table>

Drop port 80 to v

Longest Prefix Matching

<table>
<thead>
<tr>
<th>Destination</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.1.0/24</td>
<td>v</td>
</tr>
<tr>
<td>10.1.1.128/25</td>
<td>w</td>
</tr>
</tbody>
</table>

Similar approaches to handle NAT, multicast, ACLs, encapsulation, MPLS label swapping, OpenFlow, etc.

\[ P(u,v) = IP_{\text{dest}} \in 10.1.1.0/24 \]
\[ ^\wedge \text{Port}_{\text{dest}} \neq 80 \]

\[ P(u,v) = IP_{\text{dest}} \in 10.1.1.0/24 \]
\[ ^\wedge IP_{\text{dest}} \notin 10.1.1.128/25 \]
Automating Hypothesis Testing

• Could directly extend existing techniques (e.g., SAT solvers)
  – Problem: not very scalable

• Alternative solution: represent and test Boolean functions as graph traversals

• Main idea:
  – Represent network state as a forwarding graph
  – Translate hypothesis tests into graph traversals
Limiting the Search Space

Hypothesis Testing Engine

Equivalence class:
Packets experiencing the same forwarding actions throughout the network.

Generate Equivalence Classes

Updates

- 0.0.0.0/1
- 64.0.0.0/3
- 0.0.0.0/0

Fwd’ing rules

Equiv. classes
Limiting the Search Space

Hypothesis Testing Engine

Generate Equivalence Classes

Generate Forwarding Graphs

Updates

Forwarding graphs:

All the info to answer hypotheses
Limiting the Search Space

Hypothesis Testing Engine

Generate Equivalence Classes

Generate Forwarding Graphs

Run Experiments

Updates

Correct Hypotheses

Incorrect Hypotheses

Result report

• Experimental step that violates hypothesis
• Affected set of packets

Black holes, routing loops, isolation of multiple VLANs, access control policies
Evaluation

• Simulated an IP network using a Rocketfuel topology
  – Replayed Route Views BGP traces
  – 172 routers, 90K BGP updates
  – Microbenchmarked each phase of HTE’s operation
Single-Hypothesis Testing Speed

97.8% of experiments concluded within 1 millisecond.
Dealing with System Dynamics

• Challenge: Networks are Dynamic and Nondeterministic
  – May not always know what will happen given an input
  – May not always have up to date state
  – May not be fully deployed

• Solution approach: dealing with “uncertainty”
  – Explicitly model uncertainty in network’s current state
Motivating example

Case 1: update received

Case 2: update not received

Should I send B with nh=C now?

I want to shift traffic from S1 to S2.

Change your next hop to C

Change your next hop to S2

nh=C

nh=S2
Uncertainty-aware modeling: Approach

1. Derive possible network states, given inputs
2. Represent possible states using symbolic "uncertainty graph"
3. Traverse graph to test hypotheses
4. Update graph as information comes in
   - Network changes, acks from network, certain delays pass
Technical approach

Controller
- Update

Stream of Updates

GCC
- Pending Updates
- Update

Network Model
- Pass

Analysis Engine

Fail

Confirm

Update

Update

Update

Fail

22
Hypothesis Testing Time in Dynamic Networks

80% of the hypotheses tested within 10 microseconds
Conclusion

• We are constructing a hypothesis testing engine for SoS
  – Analysis methodology for reasoning about science of security of networks
  – Adds to theoretical underpinnings of SoS, supports practice of SoS

• Early results indicate feasibility
  – Experiments run in milliseconds on complex networks

• Interested in working with you
  – My contact info: caesar@illinois.edu