Automated Attack Discovery in TCP Congestion Control using a Model-guided Approach

Samuel Jero[1], Endadul Hoque[2], David Choffnes[3], Alan Mislove[3] and Cristina Nita-Rotaru[3]

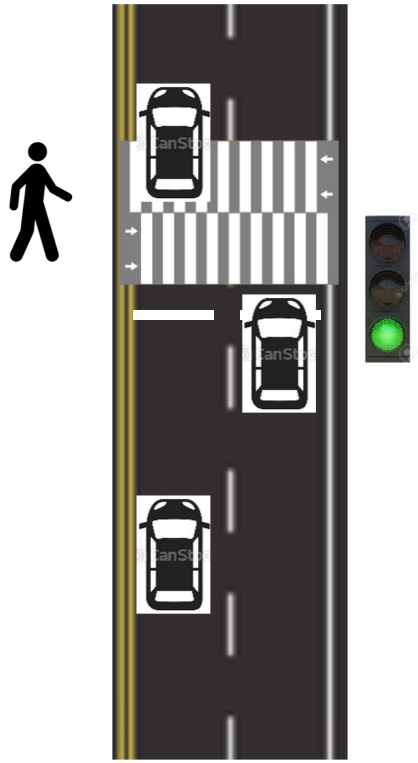
[1]Purdue University, [2]Florida International University and
[3]Northeastern University

NDSS 2018

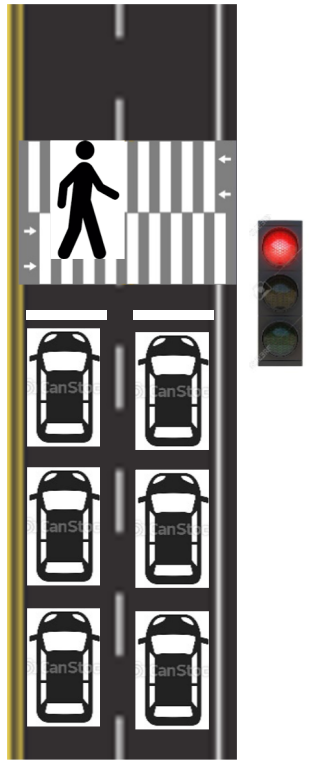
Presenter: Seong-Joong KIM

Introduction

- Assuming when you are in traffic road



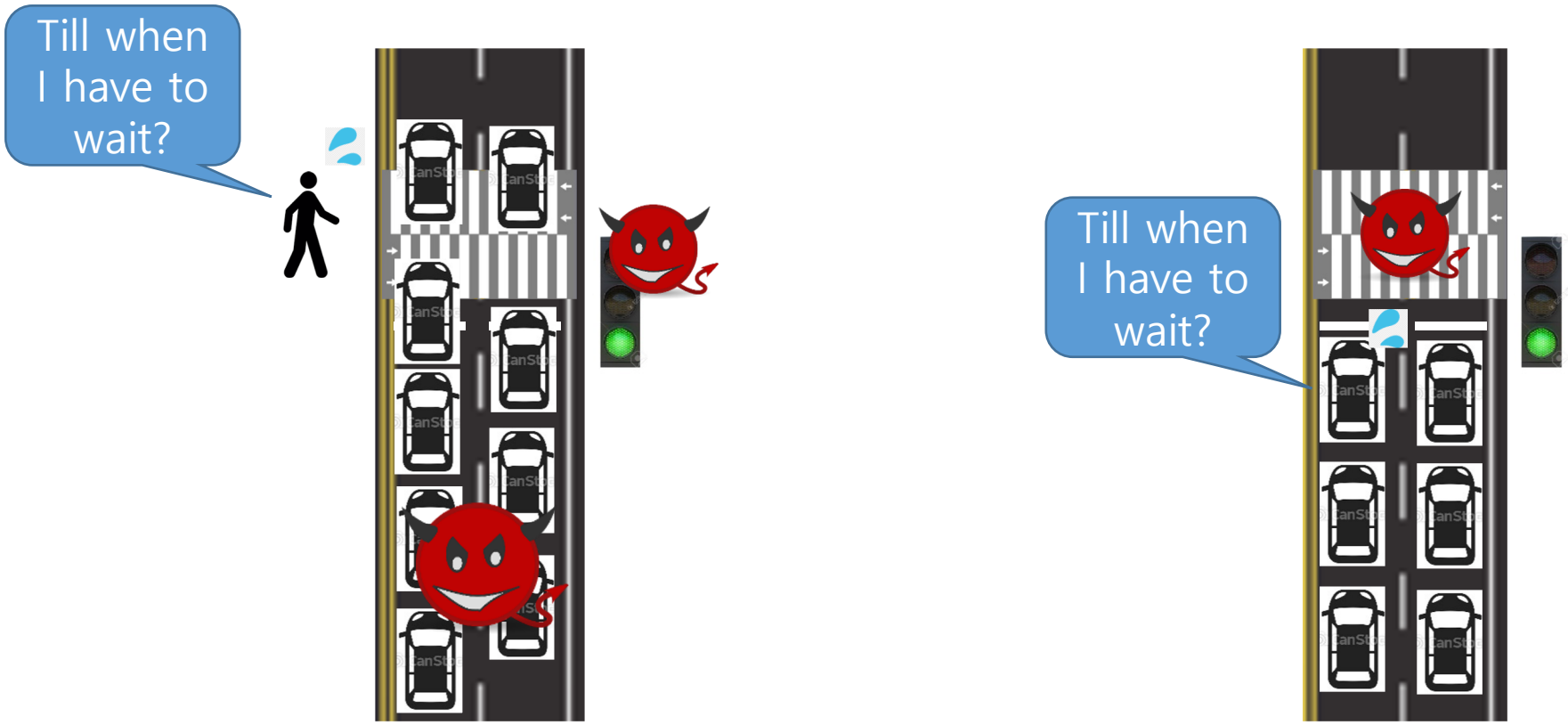
Green light



Red light

Introduction

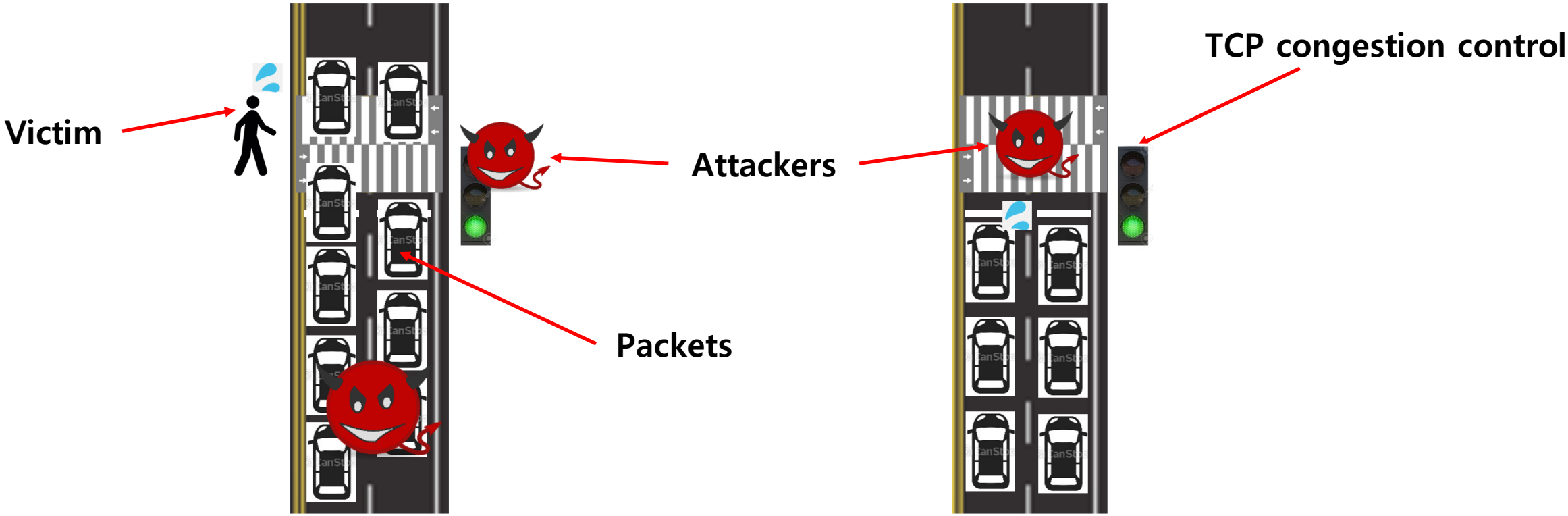
- Assuming when you are in traffic road



Causing traffic jam and delay

Introduction

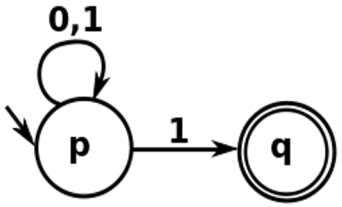
- TCP congestion control attack



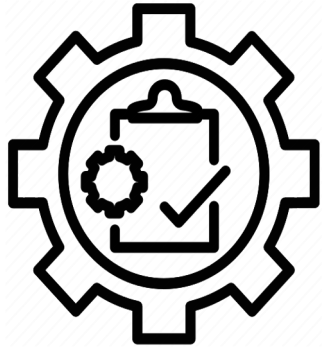
Causing throughput increasing or decreasing abnormally

Introduction

TCP congestion control



State machine



Automation Tool

Found 11 classes of TCP congestion control attacks

Background - TCP

- Used in the vast majority of Internet traffic
  - Instant messaging services, email, etc.
  - Innumerable implementations

- Features
  - Reliable data transfer, In-order delivery
  - Flow control
  - Congestion control

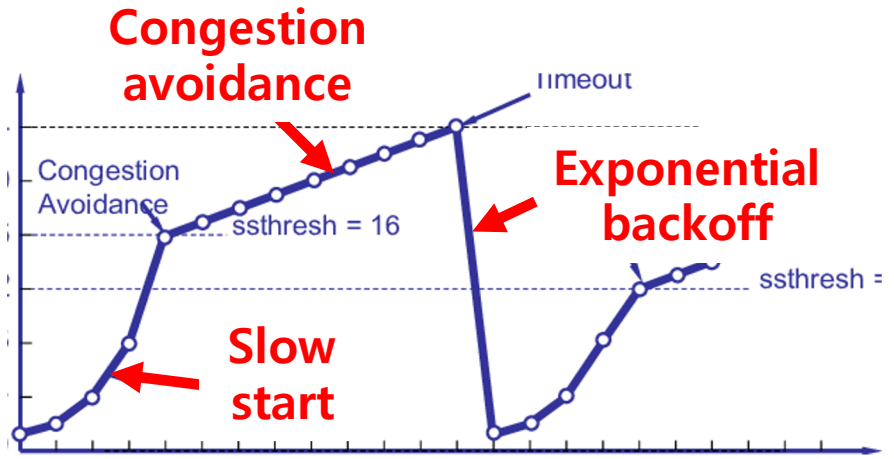
NETFLIX

Background - TCP Congestion Control

- Protects against congestion collapse
  - Majority of sent data is dropped later on
  - Caused throughout decrease of 1000x

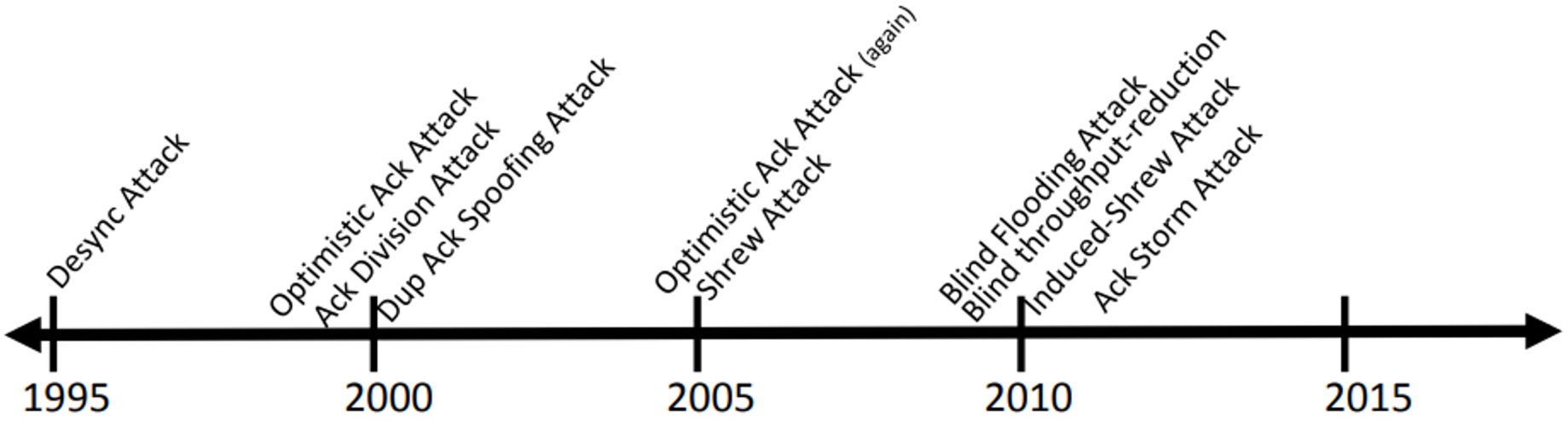
- Also ensures fairness between competing flows
  - Prevents one flow from starving others

- General scheme
  - Additive increase / multiplicative decreasing
  - Slot start, congestion avoidance
  - Fast recovery, Exponential backoff



History of TCP Congestion control Attacks

- Attacks result in
  - Increased throughput
  - Decreased throughput
  - Stalled data transfer



Why so Many Attacks?

- Attacks leverage designed behavior
  - Cause confusion of congestion control about network conditions
- Many designs and implementations
  - Variations & optimizations; hundreds of implementations
- Lack of unified specification
  - Individual components and optimizations are specified separately
- Very dynamic behavior
  - Congestion control state changes with every ACK

Current Testing Methods

- Manual Investigation
  - Manually investigate possible attacks
- Regression Testing
  - Manually create tests for known attacks
  - Test each implementation for vulnerability
- MAX – ACM SIGCOMM ’11
  - Automatically find manipulation attacks on network protocols
  - Leverage symbolic execution to identify manipulations
- SNAKE – IEEE DSN ’15 (*Best Paper Awarded)
  - Fuzz transport protocols searching for availability and performance attacks
  - Use state-machine attack injection for scalability

Proposed - TCPwn

- **Automatically test TCP implementations for congestion control attacks**
  - Test real unmodified implementations
    - No instrument or modify the implementations
  - Scalability
    - Attacks are complex and multi-stage
    - System is highly dynamic

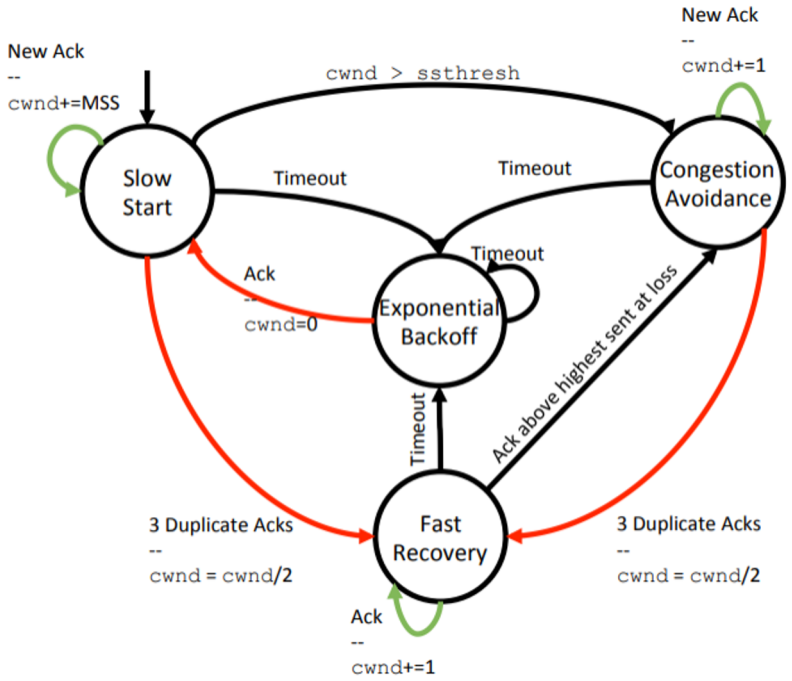
- **Approach - Modeling congestion control as a state machine**
  - Use model-based testing to identify all possible attacks in a scalable manner
  - Create testable attacks using packet manipulation and injection
  - Find attacks causing:
    - Increased & decreased throughput, or connection stalling

Example: Optimistic ACK Attack

- **Increase sending rate by ACKing data that has not been received yet**
  - Causes a denial-of-service

- **Methodology**
  - ACKing new data causes green transitions to be taken
  - Increases cwnd and thus throughput with each loop
  - Avoids red transitions which reduce cwnd and thus throughput

- **Takeaways**
  - Attacks attempt to cause desirable transitions
  - Attacks must repeatedly execute transition to have noticeable impact



TCP Congestion Control State Machine

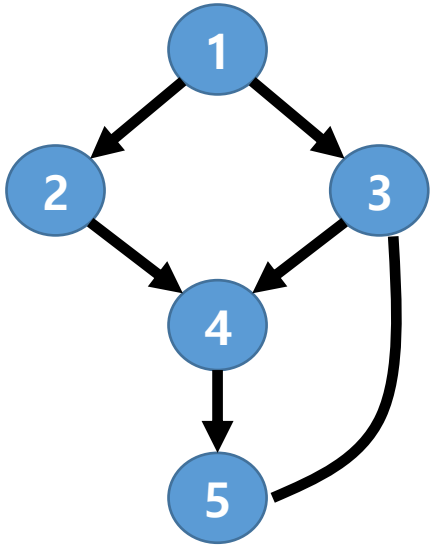
Model-based Attack Generation

Attacks generate all cycles with the following pattern
- cwnd increases/decreases along cycle
- A set of actions exist that force TCP to follow this cycle

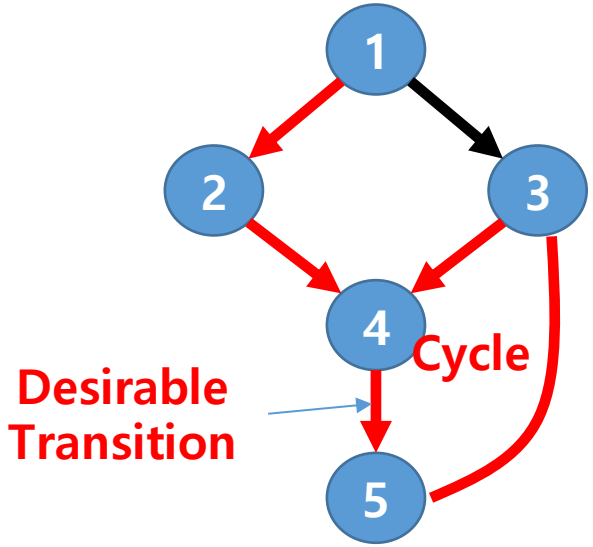
1. Consider state machine model of congestion control
2. Identify cycles containing desirable transitions
   - Abstract strategy generation
3. Force TCP to follow each cycle
   - Concrete strategy generation

Abstract Strategies

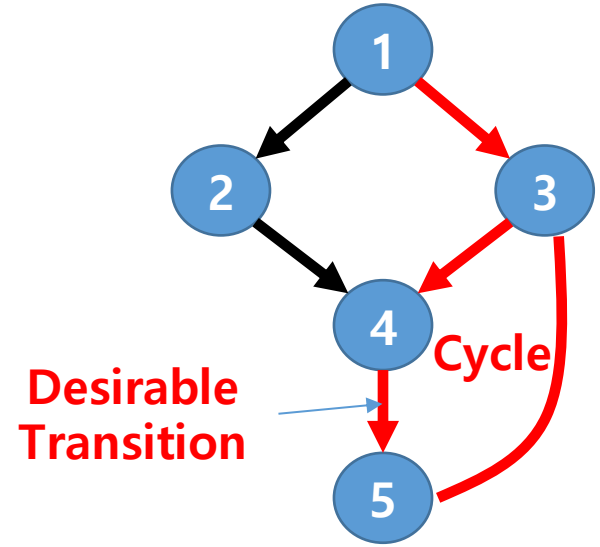
- Enumerate all paths
  - Adapt depth-first search to this problem
- Check that path contains cycle & desirable transition
  - Any change to cwnd
- Add path and transition conditions to abstract strategies



State machine



First abstract strategy

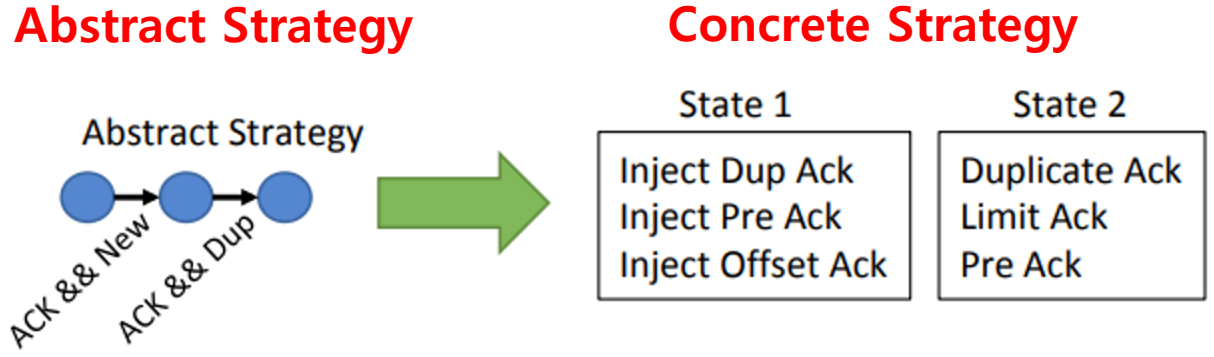


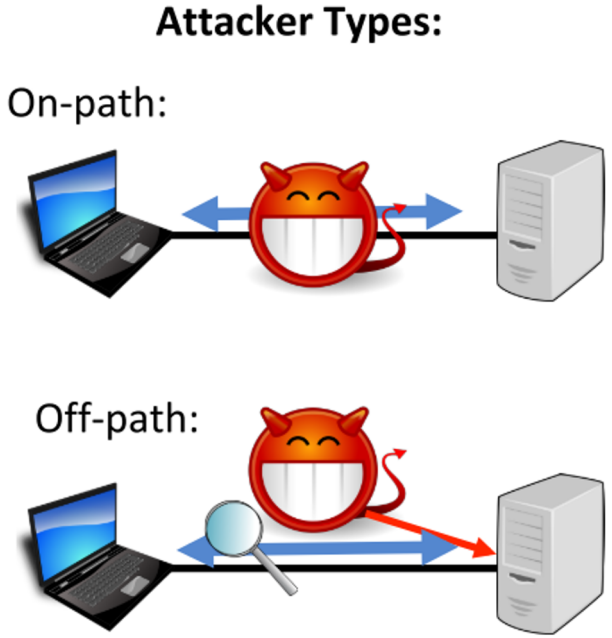
Second abstract strategy

...

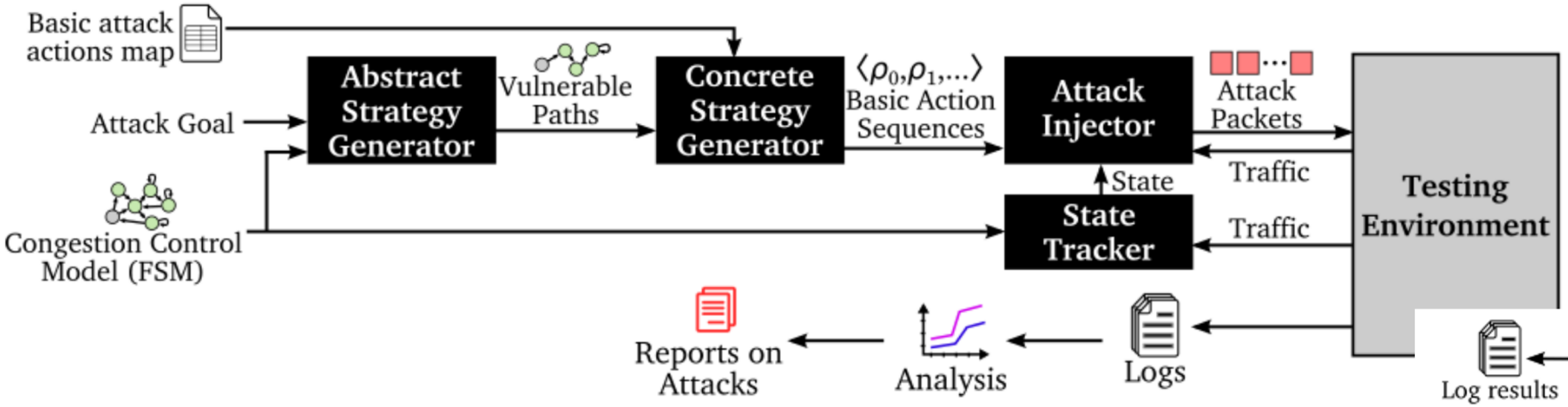
Concrete Strategies

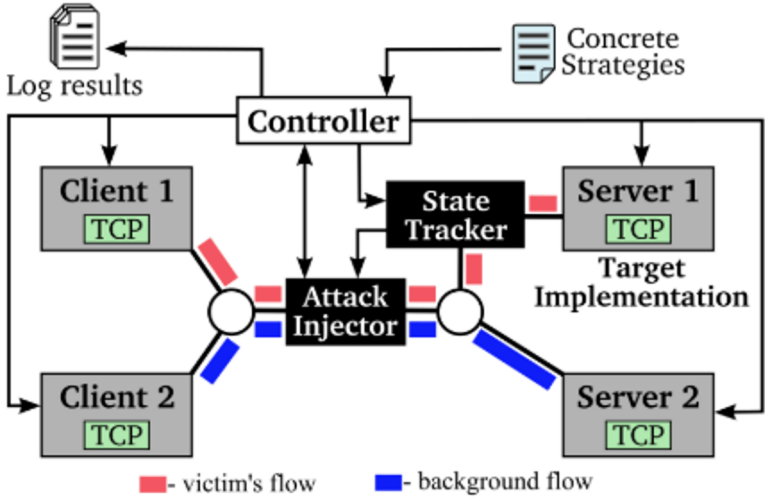
- Consider each abstract strategy separately
- Map each transition to a set of basic malicious actions
  - Actions chosen to cause transition
  - Based on attacker capabilities; on-path or off-path attackers





TCPwn Design





Testing Environment

- Abstract & Concrete Strategy Generators
  - Create strategies for model-based testing
- Attack Injector
  - Applies malicious actions
- State Tracker
  - Track the sender's congestion control state
- Run real implementations in a testbed network

Evaluation

- Tested five TCP implementations:

Implementation	Congestion Control
Ubuntu 16.10 (Linux 4.8)	CUBIC+SASCK+FRTO+ER+PRR+TLP
Ubuntu 14.04 (Linux 3.13)	CUBIC+SASCK+FRTO+ER+PRR+TLP
Ubuntu 11.10 (Linux 3.0)	CUBIC+SASCK+FRTO
Debian 2 (Linux 2.0)	New Reno
Windows 8.1	Compound TCP + SACK

Results Summary

- Found 11 classes of attacks, 8 of them unknown

Attack	Attacker	Impact	OS	New?
Optimistic Ack	On-path	Increased Throughput	ALL	No
On-path Repeated Slow Start	On-path	Increased Throughput	Ubuntu 11.10, Ubuntu 16.10	Yes
Amplified Bursts	On-path	Increased Throughput	Ubuntu 11.10	Yes
Desync Attack	Off-path	Connection Stall	All	No
Ack Storm Attack	Off-path	Connection Stall	Debian 2, Windows 8.1	No
Ack Lost Data	Off-path	Connection Stall	All	Yes
Slow Injected Acks	Off-path	Decreased Throughput	Ubuntu 11.10	Yes
Sawtooth Ack	Off-path	Decreased Throughput	Ubuntu 11.10, Ubuntu 14.04, Ubuntu 16.10, Windows 8.1	Yes
Dup Ack Injection	Off-path	Decreased Throughput	Debian 2, Windows 8.1	Yes
Ack Amplification	Off-path	Increased Throughput	Ubuntu 11.10, Ubuntu 14.04, Ubuntu 16.10, Windows 8.1	Yes
Off-path Repeated Slow Start	Off-path	Increased Throughput	Ubuntu 11.10	Yes

Results Summary

- New attacks
  - (Increasing Throughput) On-path Repeated Slow Start Attack
    - Repeated cycle of Slow Start, RTO, Slow Start due to fixed ACK number during Fast Recovery
  - (Decreasing Throughput) Dup ACK Injection Attack
    - Inject >= 3 duplicate acks repeatedly
  - (Connection Stall) Ack Lost Data Attack
    - Acknowledge lost data during Fast Recovery or Slow Start

Conclusion

- Proposed new model-guided testing for TCP congestion control
  - It uses the congestion control state machine to generate abstract strategies
  - It converts the strategies into concrete strategies made up of message-based actions.
- Implementation - TCPwn
  - To find attacks on real, unmodified implementations of TCP congestion control
- Evaluation
  - Test on 5 TCP implementations
  - Found 11 classes of attacks, 8 of which were previously unknown

Follow-up Studies

- aBBRate: Automating BBR Attack Exploration Using a Model-Based Approach (RAID '20)
  - Performed in the same lab and by the same authors
  - Found possible TCP congestion control attacks on BBR proposed by Google
  - Identified 5 classes of attacks; send faster, slower or stall

- Model-Agnostic and Efficient Exploration of Numerical State Space of Real-World TCP Congestion Control Implementations (NSDI ‘19)
  - Consider both congestion state and network environment parameters
  - Used the concept of guided random walk to find regions where the algorithm should never go
  - Found so many Linux TCP congestion control vulnerabilities and bugs

Questions

- Q1 (Tuan, *best question) The testing environment is simple with only 2 servers and 2 clients, the injector has already been set up as a proxy. Is this testing environment practical in the real world?
  - Assume two competing flows & spoofing attack ➔ sufficient assumptions
- Q2. (송민규) Why TCP does not have any cryptographic mechanisms to ensure authentication and integrity of the sent packets?
  - Before the encryption, we need to exchange keys
  - But, TCP is to have proceeded for end-to-end connection first
  - *TCP-ENO – a new IETF draft for TCP encryption

*TCP-ENO: https://www.hjp.at/doc/rfc/rfc8547.html

Questions

- Q3. (진영진) The paper covers Linux and Windows distributions. Would it be possible to apply TCPwn to MacOS distributions for checking TCP congestion control vulnerabilities?
  - Linux, Win10, Mac OS ➔ CUBIC
  - Might be possible to apply TCPwn to Mac OS
- Q4. (황영빈) In this paper, 8 new types of attacks were discovered. Are the vulnerabilities that caused the attack a matter of ~~specification~~ implementation or design?
  - Mainly focusing on design issues using a formal method
  - The following are focused on implementation bugs
    - Model-Agnostic and Efficient Exploration of Numerical State Space of Real-World TCP Congestion Control Implementations (NSDI '19)

Thank you