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

Samuel Jero, Endadul Hoque David Choffnes,
Alan Mislove, Cristina Nita-Rotaru

Background

- Previous work on attacks against TCP congestion control relied mainly on manual analysis
- We aim to automatically discover manipulation attacks on congestion control without requiring the user to provide any vulnerable line of code and without being dependent on specific implementation, language, or operating system characteristics

Approach

- model based testing
- finite state machine
- abstract model -> abstract attack strategies -> concrete attack strategies
- off-path / on-path

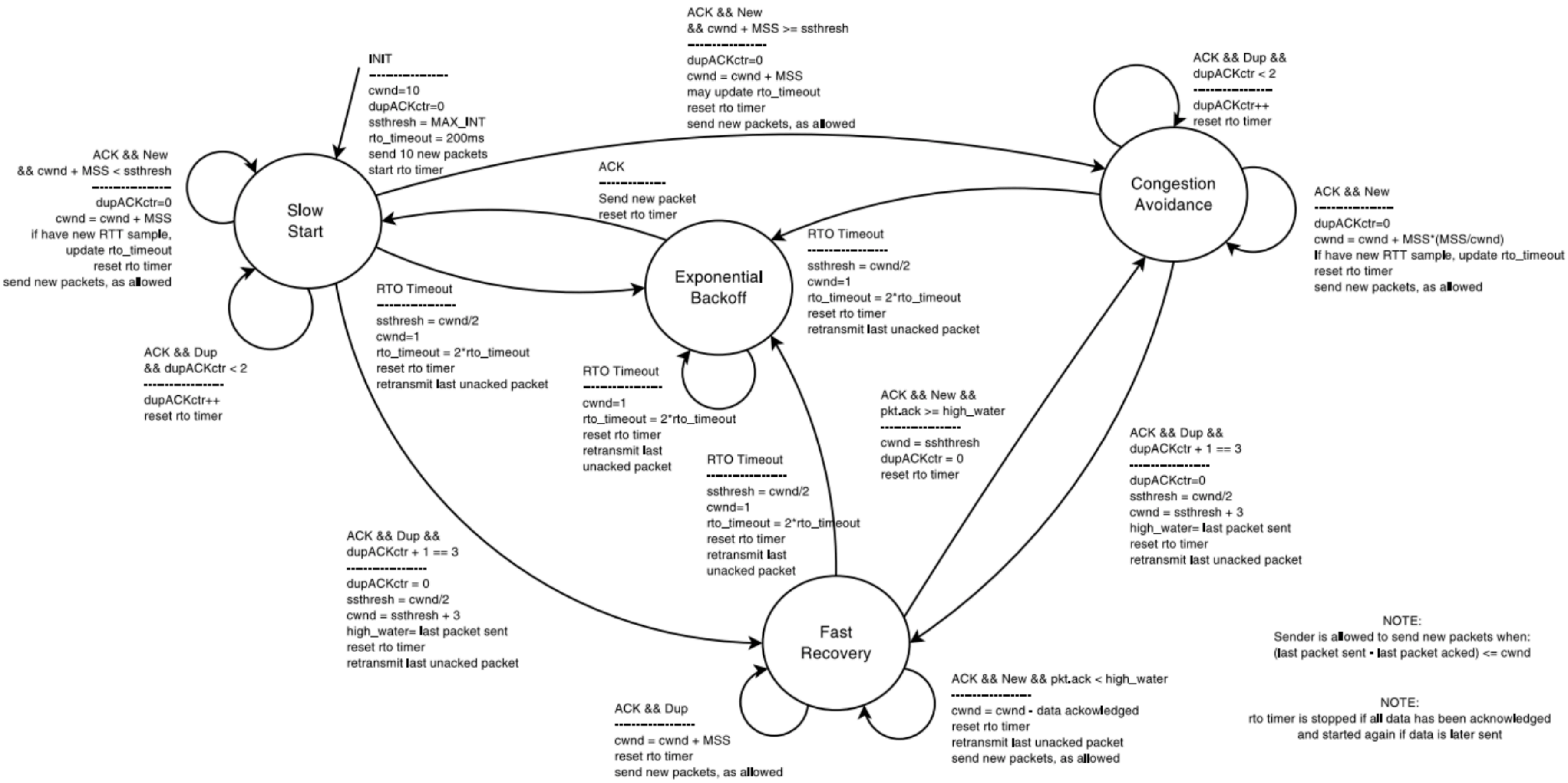
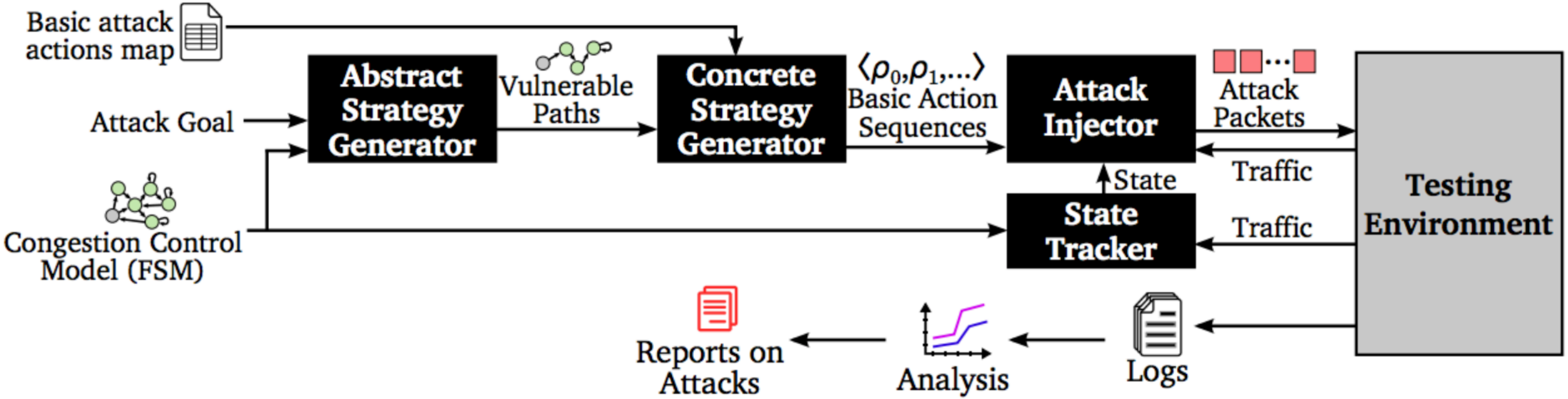


Fig. 1. TCP New Reno State Machine

Attack Model

- Attack goals
  - decreasing throughput
  - increasing throughput
  - connection stall
- Strategy and Action
  - Attacker can observe the network traffic but it doesn't have access to the source code and thus cannot instrument it

TCPWN DESIGN



Example

$\mathcal{P}$: SlowStart $\rightarrow$ FastRecovery $\rightarrow$ CongestionAvoidance $\circlearrowleft$

(**In**: SlowStart, **Condition**: `ACK && Dup && dupACKctr`$\geq$3)
(**In**: FastRecovery, **Condition**: `ACK && New && pkt.ack` $\geq$ `high_water`)
(**In**: CongestionAvoidance, **Condition**: `ACK && New`)$^{+}$

(**In**: SlowStart, **Action**: 3 $\times$ Inject `Dup-Ack`)
(**In**: FastRecovery, **Action**: Inject `Pre-Ack`)
(**In**: CongestionAvoidance, **Action**: Inject `Pre-Ack`)$^{+}$

Abstract Strategy Generation

Algorithm 1: Abstract Strategy Generator

Input: Multigraph $\mathcal{M} = (\mathcal{S}, \mathcal{N}, \mathcal{V}, \mathcal{C}, \mathcal{A}, \sigma, \mathcal{T})$, ψ, ξ, λ, $\aleph$ and a vulnerable action $\alpha \in \mathcal{A}$

Output: All vulnerable paths with respect to α

$VulnerablePaths := \emptyset$
`/* to store all the vulnerable paths */`

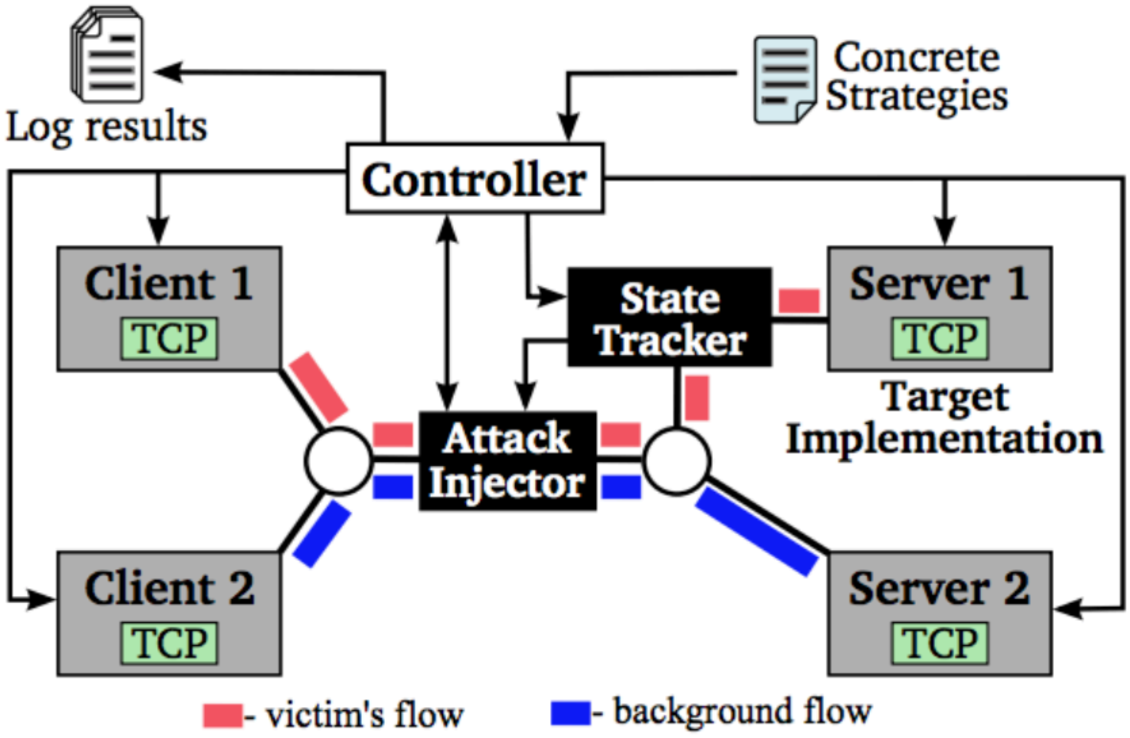
Function `VulnerablePathFinder(`$\mathcal{M}$, α`)`
- $root := \sigma$ `/* initial state */`
- Mark *root* as visited
- **foreach** *transition t such that* $\psi(t) = root$ **do**
  - Create a new path P
  - $P := P \| (root, t)$ `/* concatenating */`
  - $v := \xi(t)$
  - `RecursiveSearch(`v, P, α`)`
- **return** $VulnerablePaths$

Function `RecursiveSearch(`v, P, α`)`
`/* search continue from ` v ` */`
- $base\_case :=$ false
- **if** *v is already* visited **then** `// reached a cycle`
  - $base\_case :=$ true
- **else if** *exists no t such that* $\psi(t) = v$ **then**
  `/* ` v ` is a terminating state */`
  - $base\_case :=$ true
  - $P := P \| (v, \perp)$ `/* concatenating */`
- **if** *base_case is* true **then**
  - **if** *P is a vulnerable path w.r.t. α* **then**
    - $VulnerablePaths := VulnerablePaths \cup P$
- **else**
  - Mark v as visited
  - **foreach** *transition t such that* $\psi(t) = v$ **do**
    - $v' := \xi(t)$
    - $P' := P$ `/* creating a copy */`
    - $P := P \| (v', t)$ `/* concatenating */`
    - `RecursiveSearch(`v', P', α`)`
  - Mark v as unvisited
- **return** `/* void */`

Concrete Strategy Generation

- Injection of acknowledgements(off-path attacker)
  - Duplicate Acknowledgements
  - Offset Acknowledgements
  - Incrementing Acknowledgements
- Modification of acknowledgements(on-path attacker)
  - Division
  - Duplication
  - Pre-acknowledging
  - Limiting

Implementation



Evaluation

Num	Attack	Attacker	Description	Impact	Impl	New
1	Optimistic Ack	On-path	Acking data that has not been received	Increased Throughput	ALL	No [37]
2	On-path Repeated Slow Start	On-path	Repeated cycle of Slow Start, RTO, Slow Start due to fixed ack number during Fast Recovery	Increased Throughput	U(buntu)16.10, U11.10	Yes
3	Amplified Bursts	On-path	Send acks in bursts, amplifying the bursty nature of TCP	Increased Throughput	U11.10	Yes
4	Desync Attack	Off-path	Inject data to desynchronize sequence numbers and stall connection	Connection Stall	ALL	No [22]
5	Ack Storm Attack	Off-path	Inject data into both sides of connection, creating ack loop	Connection Stall	D(ebian)2, W(indows)8.1	No [2]
6	Ack Lost Data	Off-path	Acknowledge lost data during Fast Recovery or Slow Start	Connection Stall	ALL	Yes
7	Slow Injected Acks	Off-path	Inject acks for little data slowly during Congestion Avoidance	Decreased Throughput	U11.10	Yes
8	Sawtooth Ack	Off-path	Send incrementing acks in Congestion Avoidance/Fast Recovery, but reset on entry	Decreased Throughput	U16.10,U14.04, U11.10, W8.1	Yes
9	Dup Ack Injection	Off-path	Inject $>=$ 3 duplicate acks repeatedly	Decreased Throughput	D2, W8.1	Yes
10	Ack Amplification	Off-path	Inject acks for lots of new data very rapidly during Congestion Avoidance or Slow Start	Increased Throughput	U16.10,U14.04, U11.10, W8.1	Yes
11	Off-path Repeated Slow Start	Off-path	Repeated cycle of Slow Start, RTO, Slow Start due to increased duplicate ack threshold	Increased Throughput	U11.10	Yes