Bad Packets Come Back, Worse Ones Don't

Petros Gigis, Mark Handley, Stefano Vissicchio

University College London

SIGCOMM 2024
— SYDNEY —

AS1 provider
AS3
AS1
AS2
AS1 customer

AS1 provider
AS3
AS1
AS2
Source IP in AS2
AS1 customer

AS1 provider
AS3
AS1
Source IP in AS2
AS2
Source IP in AS2
AS1 customer

AS1 provider
AS3
AS1
Count Packets
e.g., sFlow, NetFlow
Source IP in AS2
AS2
Source IP in AS2
AS1 customer

AS1 provider
Source IP in AS2
Spoofed Traffic
AS3
AS1
Count Packets
e.g., sFlow, NetFlow
Source IP in AS2
Legitimate Traffic
AS2
Source IP in AS2
Legitimate Traffic
AS1 customer

How can we distinguish which case we are in?

How can we distinguish which case we are in?

What is legitimate traffic?

— **Traffic that originates where it claims to originate from.**

How can we distinguish which case we are in?

What is legitimate traffic?

— **Traffic that originates where it claims to originate from.**

How to detect legitimate traffic?

How to detect legitimate traffic?

- Hard to distinguish from spoofed traffic.

How to detect legitimate traffic?

- Hard to distinguish from spoofed traffic.
- Expected to respond to feedback in a closed-loop communication.

How to detect legitimate traffic?

- Hard to distinguish from spoofed traffic.
- Expected to respond to feedback in a closed-loop communication.

💡 **Closed-loop traffic** can be used as **a proxy to detect legitimate traffic.**

How to detect legitimate traffic?

- Hard to distinguish from spoofed traffic.
- Expected to respond to feedback in a closed-loop communication.

💡 **Closed-loop traffic** can be used as **a proxy to detect legitimate traffic.**

How to detect closed-loop traffic?

— Tweak traffic.

How to detect legitimate traffic?

- Hard to distinguish from spoofed traffic.
- Expected to respond to feedback in a closed-loop communication.

💡 **Closed-loop traffic** can be used as **a proxy to detect legitimate traffic.**

How to detect closed-loop traffic?

— Tweak traffic.

How to detect legitimate traffic?

- Hard to distinguish from spoofed traffic.
- Expected to respond to feedback in a closed-loop communication.

💡 **Closed-loop traffic** can be used as **a proxy to detect legitimate traffic.**

How to detect closed-loop traffic?

— Tweak traffic.

How to detect legitimate traffic?

- Hard to distinguish from spoofed traffic.
- Expected to respond to feedback in a closed-loop communication.

💡 **Closed-loop traffic** can be used as **a proxy to detect legitimate traffic.**

How to detect closed-loop traffic?

— Tweak traffic.

How to detect legitimate traffic?

- Hard to distinguish from spoofed traffic.
- Expected to respond to feedback in a closed-loop communication.

💡 **Closed-loop traffic** can be used as **a proxy to detect legitimate traffic.**

How to detect closed-loop traffic?

— Tweak traffic.

How to detect legitimate traffic?

- Hard to distinguish from spoofed traffic.
- Expected to respond to feedback in a closed-loop communication.

💡 **Closed-loop traffic** can be used as **a proxy to detect legitimate traffic.**

How to detect closed-loop traffic?

— Tweak traffic.

How to detect legitimate traffic?

- Hard to distinguish from spoofed traffic.
- Expected to respond to feedback in a closed-loop communication.

💡 **Closed-loop traffic** can be used as **a proxy to detect legitimate traffic.**

How to detect closed-loop traffic?

— Tweak traffic.

TCP is the perfect candidate.

What is the easiest way to tweak TCP traffic?

What is the easiest way to tweak TCP traffic?

— Drop a data packet.

Is dropping one packet enough?

- So, we drop a data packet:
  - If a **retransmission is observed**, the **flow is closed-loop**.
  - If **no retransmission** is observed, the **flow is not closed-loop (spoofed).**

Is dropping one packet enough?

- So, we drop a data packet:
  - If a **retransmission is observed**, the **flow is closed-loop**.
  - If **no retransmission** is observed, the **flow is not closed-loop (spoofed).**

- **What could go wrong?**

Is dropping one packet enough?

- So, we drop a data packet:
  - If a **retransmission is observed**, the **flow is closed-loop**.
  - If **no retransmission** is observed, the **flow is not closed-loop (spoofed).**

- **What could go wrong?**

When closed-loop —>

Is dropping one packet enough?

- So, we drop a data packet:
  - If a **retransmission is observed**, the **flow is closed-loop**.
  - If **no retransmission** is observed, the **flow is not closed-loop (spoofed).**

- **What could go wrong?**

When not closed-loop —>

The signal from **a single** data packet drop is **weak and noisy**!

How can we improve this?

The signal from **a single** data packet drop is **weak and noisy**!

How can we improve this?

Drop **a few** data packets to **gain confidence.**

(The) **Penny** drops

- **Approach:** Statistical model **comparing two competing hypotheses:**
  - **H1**: hypothesis that the flow **is closed-loop**.
  - **H2**: hypothesis that flow **is not closed-loop (spoofed).**

(The) **Penny** drops

- **Approach:** Statistical model **comparing two competing hypotheses:**
  - **H1**: hypothesis that the flow **is closed-loop**.
  - **H2**: hypothesis that flow **is not closed-loop (spoofed).**

Counters & Parameters

n_{RTX}: number of observed retransmissions for packets we dropped

n_{noRTX}: number of packets we dropped for which we didn't observe a retransmission

p_{noRTX}: assumed probability that we do not observe a retransmitted packet within a closed-loop flow

f_{dup}: fraction of observed packets with one or more duplicates

Probabilities

$$P(H1) = (p_{noRTX})^{n_{noRTX}}$$

$$P(H2) = (f_{dup})^{n_{RTX}}$$

$$P(genuine) = P(H1)/(P(H1) + P(H2))$$

(The) **Penny** drops

- **Approach:** Statistical model **comparing two competing hypotheses:**
  - **H1**: hypothesis that the flow **is closed-loop**.
  - **H2**: hypothesis that flow **is not closed-loop (spoofed).**

(a) Not closed-loop flow (b) Closed-loop flow

Dealing with short flows

Dealing with short flows

Dealing with short flows

Problem: Not enough data packets to reach a conclusion.

Dealing with short flows

Problem: Not enough data packets to reach a conclusion.

Solution: Aggregate stats from multiple flows.
Apply the same statistical model.

Penny at Runtime: The devil's in the details

Complications:

- **Dealing with the TCP protocol**
- **Dealing with network conditions**
- **Dealing with malicious sources**

Penny at Runtime: The devil's in the details

Complications:

- Dealing with the TCP protocol
- Dealing with network conditions
- Dealing with malicious sources

Mechanisms:

- **Selection of packets to drop**
- **Counter snapshots**
- **Conservative thresholds and parameters**

Waiting for retransmissions

Waiting for retransmissions

Waiting for retransmissions

Waiting for retransmissions

Waiting for retransmissions

Waiting for retransmissions

Waiting for retransmissions

Dealing with interrupted flows

Dealing with interrupted flows

Dealing with interrupted flows

Dealing with interrupted flows

Dealing with interrupted flows

Dealing with duplicates

TCP Flow

Dealing with duplicates

Dealing with duplicates

Dealing with duplicates

Dealing with duplicates

- We treat flows with 15% loss as suspicious.
- Rely on stats to cope with < 15% dups.

Does it work?

- Evaluation with NS-3:
  - **Multiple TCP variants**: NewReno, Cubic, ...
  - **Diverse network conditions**: upstream/downstream losses, queues, ...
  - **Varied input traffic**: closed-loop, worst-case not closed-loop, mixed traffic, short/long flows, ...
  - **Different Penny parameters**: packet drop rate, timers, ...

For aggregates with only closed-loop traffic, Penny's **stats** always work

For malicious traffic, Penny's **stats** work whenever we drop enough packets

For malicious traffic, Penny's **stats** work whenever we drop enough packets

For malicious traffic, Penny's **stats** work whenever we drop enough packets

For malicious traffic, Penny is always correct as each drops at least 12 packets

For mixed traffic, Penny's **stats** do not always work

For mixed traffic, Penny's **stats** do not always work, but **Penny** does

Penny switches to test (some) individual flows when aggregates look spoofed

Other results

- Flow performance degradation is negligible.
  - We only drop 12 carefully selected packets per test.

- Feasibility of system implementation.
  - Low processing requirements
  - Low memory requirements.

Takeways

- We presented **Penny**: a **robust**, **reliable**, and **practical traffic checker** that interacts with transiting packets to assess if they are part of legitimate, and hence non-spoofed TCP flows.

- We have shown that by **dropping a few TCP packets**, we can extract a **strong signal on the type of TCP traffic.**

- Targeted drops cause **negligible impact on the performance of flows**, especially in the case of aggregates.

- Using **Penny as a network primitive** to enhance the accuracy of ISPs' measurement and monitoring systems.

Backup Slides

Penny's impact on individual flows

- Experiment setup:
  - TCP background traffic
  - 1 MB-long Cubic flows
- Dropping with a 5% probability (12 drops) leads to a faster conclusion and has the same impact as a 1% random loss.
- Similar results for other TCP variants.

Penny's impact on flow aggregates

- Experiment setup:
  - TCP background traffic
  - 100 non-spoofed TCP flows

- Penny has a **negligible impact** on TCP flow completion times when running **on aggregates**.

Accuracy of **Penny**'s statistical model

(a) 20% closed-loop - 80% spoofed

(b) 10% closed-loop - 90% spoofed

Penny's statistical model

Hypotheses

H1: hypothesis that the flow under test is closed-loop

H2: hypothesis that the flow under test is not closed-loop

Parameters

p_{drop} : probability of dropping a TCP data packet

p_{noRTX} : probability miss a retransmission within a closed-loop flow

Measurement counters

n_{RTX} : # of observed retransmissions for packets we dropped

n_{noRTX} : # of packets we dropped for which we did not observe a retransmission

f_{dup} : fraction of observed packets with one or more duplicates

Probabilities

$$P(\text{H1}) = (p_{\text{noRTX}})^{n}\text{noRTX}$$

$$P(\text{H2}) = (f_{\text{dup}})^{n}\text{RTX}$$

$$P(\text{genuine}) = P(H1)/(P(H1) + P(H2))$$

Procedure

$$P(\text{genuine}) > 0.99 \implies \text{closed-loop}$$

$$P(\text{genuine}) < 0.01 \text{ or } f(\text{dup}) > 0.15 \implies \text{not closed-loop}$$