NetWarden: Mitigating Network Covert Channels while Preserving Performance

Jiarong Xing, Qiao Kang, Ang Chen
Rice University
Motivation: Mitigating network covert channels

Launch code: 1011

President

Secretary

Launch code

Code is 1011

Attacker

Ah! 1011
Motivation: Mitigating network covert channels

- Covert channels:
  - Storage channels: changing the packet header fields.

Launch code: 1011

President
Secretary

TCP hdr: 1011
HELLO

Ah! 1011
Attacker
Motivation: Mitigating network covert channels

• Covert channels:
  • Storage channels: changing the packet header fields.
  • Timing channels: changing the timing of packets.
Covert channels are a difficult problem

- Detection is difficult due to non-determinism
  - Protocols fields, e.g., TCP initial sequence numbers
  - Protocol events, e.g., inter-packet gaps

- Mitigation is difficult due to high traffic volume
  - Operate on every packet of Tbps traffic
Many defenses have been proposed:

- **ISN storage channel:**
  - Add the same offset to TCP sequence numbers
- **IP timing channel:**
  - Add random delays to destroy original timing
Common problem: Performance penalty

- Per-packet operation running in software is very inefficient
  - Switches have to handle Tbps traffic
- Incur extra delay to TCP connections
  - Many flows are delay sensitive
- Not practical for deployment!
Key question

Can we mitigate covert channels while *preserving performance*?
New opportunity: Programmable switches

- Programmable switches
  - Can be programmed in high-level languages (e.g., P4)
  - Run at linespeed (Tbps)
  - Widely used in network community:
    - SilkRoad-Sigcomm’17, Speedlight-Sigcomm’18, Blink-NSDI’19, Contra-NSDI’20, etc.
  - An ideal candidate for covert channel defenses!
NetWarden: A programmable switch defense

- NetWarden mitigates covert channels while preserving performance.
Key features

Effective defenses
Defend against a wide range of covert channels

High throughput
~100Gbps per switch port

Performance preservation
< 1% performance deviation

Great transparency
Totally transparent to end hosts

NetWarden
Outline

- Motivation: Mitigating network covert channels
- State of the art: Performance penalty
- Approach: NetWarden
  - NetWarden design
    - Challenge #1: Limited programming model
    - Challenge #2: Performance preservation
- Evaluation
- Conclusion
Challenge #1: Limited programming model

- **Solution**: hardware/software co-design principle.
Programmable switch anatomy

- Data plane
  - ☹️ Header modification, ns timestamp, per-flow state, line speed
  - 😞 Limited memory, limited arithmetic computation

- Control plane
  - ☹️ Abundant memory, complex arithmetic computation
  - 😞 Software speed, make the circle disappear
Applying the hardware/software co-design principle

Switch control plane

- Computation: Batch operations
- Memory: Growing state
- Minimize crosstalk

Switch data plane

- Computation: Per-packet operations
- Memory: Constant state
Applying the hardware/software co-design principle

Switch control plane

Packet caching

Statistical tests

Switch data plane

RTT measurement

Per-flow state maintenance

IPD maintenance

Header modification
Challenge #2: Performance preservation

- Problem: Existing mitigations incur performance loss.
- Takeaway: Create the illusion of the same RTT for the sender.
Solution: Performance boosters

- **ACK booster**
  - Maintain the same RTT by generating ACK packets in advance.
- **Receiving window booster**
  - Send more data per RTT by enlarging the receiving window size.
Performance preservation: ACK booster

- Creates the **illusion** of the same RTT as perceived by the sender.
Outline

- Motivation: Mitigating network covert channels
- State of the art: Performance penalty
- Approach: NetWarden
- NetWarden design
  - Challenge #1: Limited programming model
  - Challenge #2: Performance preservation
- Evaluation
  - Conclusion
Evaluation setup

• NetWarden prototype:
  • Runs in Tofino Wedge 100BF-32X switch.
  • 2500 LoC of P4 + 3000 LoC of C+Python

• Threat model:
  • A compromised server + a trusted P4 switch running NetWarden
  • Leak a 2048-bit RSA key via covert channel.

• Real world applications:
  • Apache servers, FTP servers, Nodejs servers

• Baseline:
  • No defense: No covert channel defenses are deployed.
  • Naïve defense: Covert channel defenses without performance preservation
How effective is NetWarden in covert channels mitigation?

- Naïve defense: renders decoding to a random guess.
- NetWarden: very close to a random guess.
- NetWarden can mitigate covert channels **effectively**.
How well does NetWarden preserve performance?

- Naïve defense incurs 25% performance penalty.
- NetWarden only has 1% performance deviation.
- NetWarden can mitigate covert channels with minimal performance loss.
See more evaluation results in our paper

- System scalability
- System overhead
- Different TCP variants
- Complex Applications
- Self-defense techniques
Outline

- Motivation: Mitigating network covert channels
- State of the art: Performance penalty
- Approach: NetWarden
- NetWarden design
  - Challenge #1: Limited programming model
  - Challenge #2: Performance preservation
- Evaluation
- Conclusion
Conclusion

• Motivation: Mitigating network covert channels
• Key limitation of existing approaches:
  • **Performance penalty**
• Our approach: NetWarden
  • Principles of hardware/software co-design
  • Efficient defenses
  • Performance preservation
• Evaluation:
  • Mitigates covert channels with **minimum performance loss**!
• Source code: [https://github.com/jiarong0907/NetWarden](https://github.com/jiarong0907/NetWarden)
• Looking for internship in 2021 summer!

Questions?