A Fast Worm Scan Detection Tool for VPN Congestion Avoidance

Arno Wagner,Thomas Dübendorfer, Roman Hiestand,

Christoph Göldi, Bernhard Plattner

Communication Systems Group

Swiss Federal Institute of Technology Zurich (ETH Zurich)

Outline

1. Setting
2. Problem Statement
3. Design Goals
4. Approach
5. Implementation
6. Tests, Validation
7. Remarks

Setting

Collaboration between OpenSystems AG, Zurich and Communication Systems Group (CSG) at ETH Zurich

OpenSystems:

- Provides, e.g., managed VPN links on OpenSystems hardware (Linux based industrial PCs)
- Administration remotely from Zurich NOC
- Customers in 70 countries
- > 100 VPN links operational

Problem Statement

Internet links: SAT-phones, . . ., high-speed Internet

- Congestion can cause significant problems
- Remote diagnosis can be difficult
- Customers may need their links repaired very fast
- IT competence at customers sites varies strongly
- Diagnosis has to be done remotely by OpenSystems
  ⇒ Typically requires several hours of manual work

One main congestion source: Worm infected hosts

Design Goals

The scan traffic detector needs to:

- Run on the VPN endpoints (detection is for attached network, not VPN tunnel)
- Communicate only when problems are detected
- Have low resource needs
- Detect scan and DoS traffic fast
- Identify infected hosts
- Have a low false positives rate

Approach

Main approach: Failed connection counting on a per-host basis

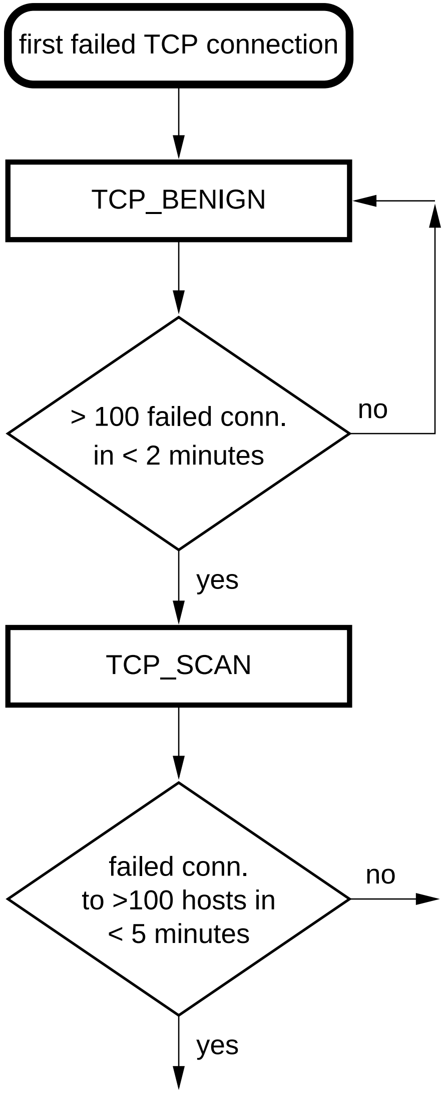
- Relatively simple for TCP
- Still works for UDP and ICMP

Idea is well documented in the literature

TCP Scan Detection

- Each active host has a state
- Measurements are done incrementally
- Identified infected hosts are ignored to reduce load

TCP Host Flowchart I



first failed TCP connection

TCP_BENIGN

> 100 failed conn. in < 2 minutes

no

yes

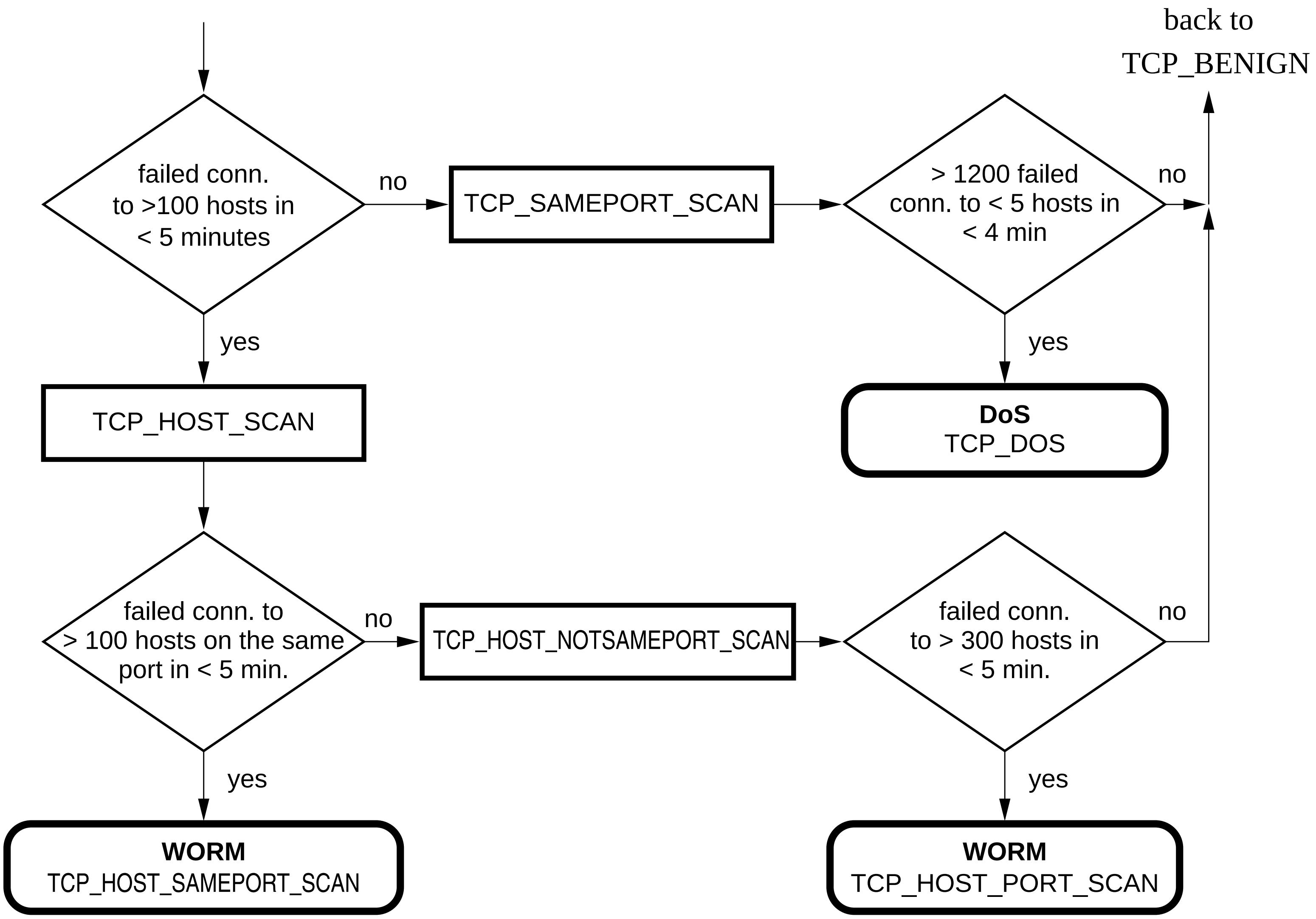
TCP_SCAN

failed conn. to >100 hosts in < 5 minutes

no

yes

TCP Host Flowchart II



back to
TCP_BENIGN
failed conn. to >100 hosts in < 5 minutes
no
TCP_SAMEPORT_SCAN
> 1200 failed conn. to < 5 hosts in < 4 min
no
yes
TCP_HOST_SCAN
yes
DoS
TCP_DOS
failed conn. to > 100 hosts on the same port in < 5 min.
no
TCP_HOST_NOTSAMEPORT_SCAN
failed conn. to > 300 hosts in < 5 min.
no
yes
yes
WORM
TCP_HOST_SAMEPORT_SCAN
WORM
TCP_HOST_PORT_SCAN

UDP Scan Detection

Similar to TCP, but failed ”connection” can be

- UDP packet, no answering packet
- UDP packet, answering ICMP ”destination unreachable”

Further differences:

- Thresholds are different
- More traffic needed to trigger detection

ICMP Scan Detection

Similar to TCP, but

- No port checks
- Failed ”connections” possibilities (not distinguished):
  - ICMP ”destination unreachable”
  - ICMP requests without answer

Detection Latency

Tests are done serially to conserve memory

+ No traffic needs to be stored
- Worst case detection time is higher

But:

- More scan traffic gives faster detection
- Worst case detection times are still reasonable (TCP: 17 min, UDP: 18 min)

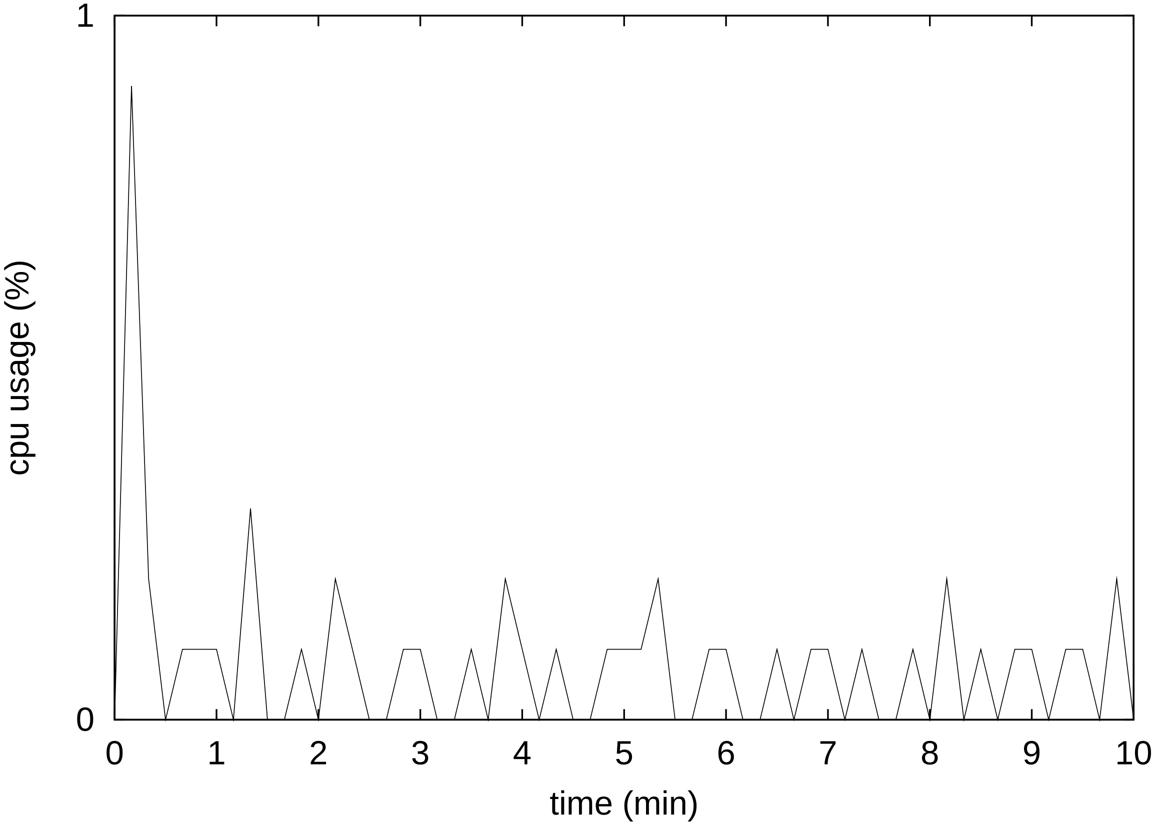
Implementation

VPN node: P4 2.4GHz, 1GB RAM, 2 * FE, 2 * GbE, customised 2.4 kernel

- Detector: Uses Bro intrusion detection system
  Traffic capturing via `libpcap`
- Event propagation via system log
- Alerting via OpenSystems log monitor

Performance I

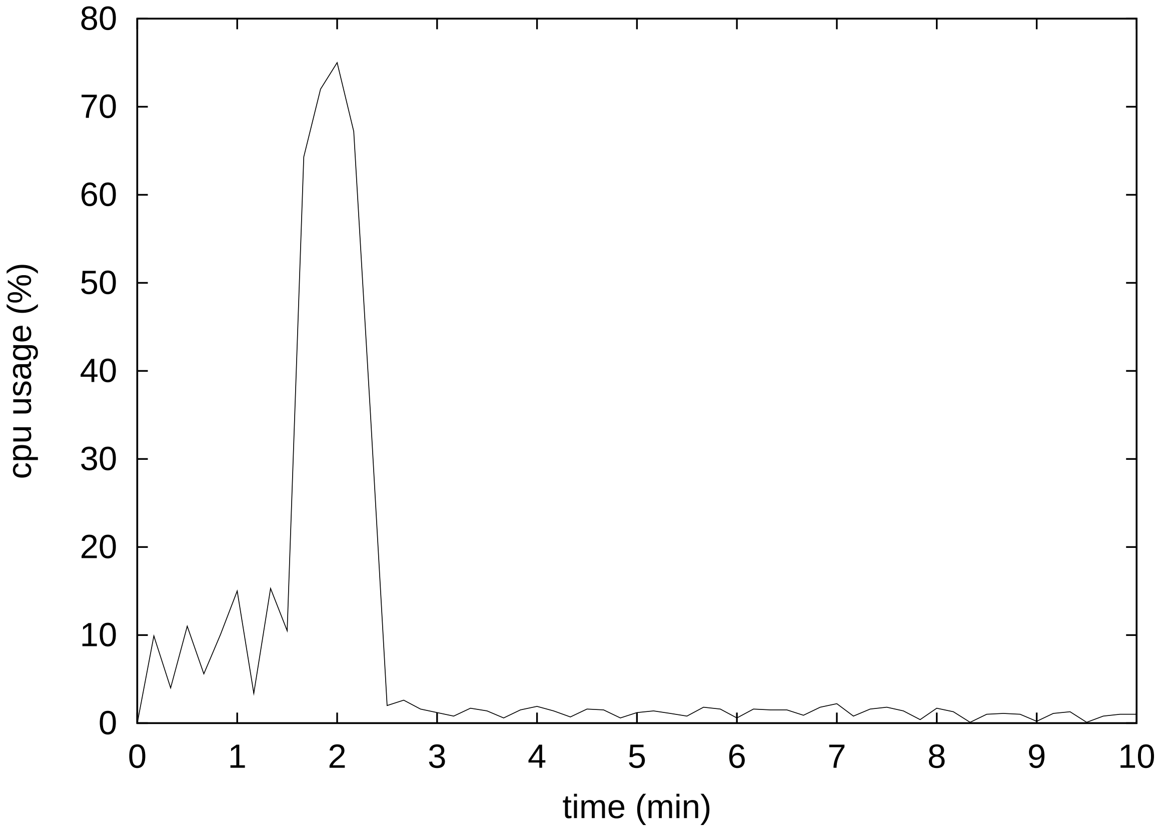
4 infected hosts, simulated TCP worm traffic (MACE)



Memory load stays below 8MB

Performance II

252 infected hosts, simulated TCP worm traffic (MACE)



Memory load stays below 20MB

Validation

The detector was tested with

- Blaster worm (real infection)
- SQL Slammer worm (real infection)
- Simulated worm traffic by MACE
- 22 hours of productive traffic from 15 VPN links
  ⇒ No false positives
- Several different P2P filesharing applications
  ⇒ Alerts from eMule when firewalled and searching

Remarks I

- Detector is used in OpenSystems production environment
- Source code available upon request:
  Detector scripts (Bro): GPL
  MACE extensions: non-commercial use
- Very successful project:
  - Students: Very good master’s thesis
  - OpenSystems: Practical solution of a real problem
  - ETH: Paper at DIMVA’06

Remarks II: Collaboration

Industrial collaboration with OpenSystems on student theses works well.

- No money flows
- Topics must be of interest to all sides
- Everybody invests real effort and shapes results
- Mutual understanding of different goals
- Produced software GPL where possible
- Important: Avoid ”problem students”

Thank You!