Tolerating Overload Attacks Against Packet Capturing Systems

Antonis Papadogiannakis  
Michalis Polychronakiss  
Evangelos P. Markatos

FORTH-ICS, Greece
Columbia University, USA
FORTH-ICS, Greece
Network Traffic Monitoring Applications

• Network-level intrusion detection
• Network traffic classification
• Next-generation firewalls

• Based on packet capturing systems
• Need to analyze all packets at real-time on multi-Gigabit links
What Happens in Case of Overload?

• Short-term traffic or processing bursts
• Handled with **limited buffering** in main memory

• When memory buffer becomes full, excess packets are *randomly* dropped

• **Important** packets may be lost
  • e.g. missed attacks
Even Worse: Overload Attacks

- Algorithmic complexity attacks
  - Exploit the differences between average case and worst case complexity
  - Low bandwidth attacks
  - Each packet results to orders of magnitude slower processing

- Traffic overload attack
  - Higher rates than the system can handle
Existing Solutions

• Provisioning
  – Tune the system for worst case scenarios

• Thresholds

• Algorithmic solutions
  – Reduce the difference between average and worst case performance

• Selective discarding
  – Discard the less important packets under overload
Our Approach: Selective Packet Paging

• Two layer memory management
  – Excess packets are buffered to secondary storage
  – All packets are analyzed

• Detection with randomized timeout interval
  – Detects malicious packets that slowdown the system
  – Send these packets to secondary storage
Packet Paging: Two-layer Memory Management

Packet Receive Index (bitmap)

M: packet stored in memory
D: packet stored in disk
Packet Paging: Two-layer Memory Management

- Modern disks
  - Three orders of magnitude more storage than main memory
  - Disk throughput increases, multiple disks organized in RAID0
- Packet capturing and storing thread has higher priority than packet processing thread
- Disk I/O thread prefetches packets from disk and optimizes throughput
**Selective Packet Paging**

- With packet paging, both benign and malicious packets are buffered to disk
  - Increased delay for benign packets

- With *selective packet paging* only malicious packets are buffered to disk
  - Benign packets do not experience any delay
  - Malicious packets experience almost the same delay
Malicious Packet Detection

• Malicious packets: packets with unusually high processing times

• The problem: detecting malicious packets that slowdown the system with minimum overhead
1st way: Using the Timestamp Counter

```c
read_timestamp();
process_packet();
read_timestamp();
```

Problem 1: the packet cannot be evicted
Problem 2: measures the total system time, not application’s CPU time

User level, not interrupt driven
2nd way: Setting a Timer per-packet

```
set_timer(interval);
process_packet();
```

Problem 1: runtime overhead for setting a timer per packet

Problem 2: a constant or predictable interval can be used by attackers to evade detection
set_timer(random interval);

process_packets();

At timer expiration: check if the same packet is being processed

The timeout interval is chosen randomly from [low, high]

papadog@ics.forth.gr

USENIX ATC 2012
Detection Probability

\[ P(\text{det}) = \frac{a \times d}{d + t} \times \frac{(d - \text{low})^2}{2 \times d \times (\text{high} - \text{low})} \]

- \( a \) is the percentage of attack packets
- \( d \) is the average processing time of attack packets
- \( t \) is the average processing time of benign packets

Probability of detecting an attack after \( N \) timeouts:

\[ 1 - (1 - P(\text{det}))^N \]

approaches 1 very fast

Average detection time:

\[ T = \frac{1}{P(\text{det})} + 1 \text{ timeouts} \]

\[ T \times (\text{high} - \text{low}) / 2 \mu s \]
Detection Time

- Attack detected within 10 ms for 100% of ACKs
- Attack detected within 170 ms for 25% of attack packets and 100 μs processing delay

\[ \text{low} = 50 \mu s, \quad \text{high} = 1000 \mu s \]
Implementation

• Within the popular libpcap packet capture library

• Existing applications can use selective packet paging without code modifications
Experimental Evaluation

- Two experiments
  - Algorithmic complexity attack against Snort’s regular expression matching using excessive backtracks
  - Traffic overload attack against Snort
Algorithmic Complexity Attack: Dropped Packets

For $10^4$ crafted packets per minute pcap loses more than 80% of the packets

Only the algorithmic attack packets are buffered to disk by SPP

No packet dropped with SPP for up to $10^6$ crafted packets per minute

Packets dropped with original pcap
Packets dropped with Selective Packet Paging
Packets buffered to disk with Selective Packet Paging
Algorithmic Complexity Attack: Detected Attacks

Snort on top of SPP detects all attacks for rates up to $10^6$ packets/minute.

Snort on top of the original pcap misses all attacks for $10^5$ crafted packets/minute.
Traffic Overload Attack: Dropped Packets

For 2 Gbit/s traffic burst the original pcap drops 53% of the packets

Pcap with SPP drops no packets for traffic bursts up to 2 Gbit/s

Dropped packets with original pcap are buffered to disk with SPP

30-seCONDS LONG TRAFFIC BURSTS
Traffic Overload Attack: Detected Attacks

30-seconds long traffic bursts

Snort with SPP misses no attack for burst rate up to 2 Gbit/sec

For 2 Gbit/s burst rate
Snort with original pcap miss 32.5% of the attacks
Summary

• Difficulty to predict and avoid overloads
• As monitoring applications become more complex, they may be more vulnerable to algorithmic attacks

Selective Packet Paging: generic defense against unknown algorithmic attacks and unpredictable overloads
  – Two-layer memory management, using secondary storage
  – Detection based on randomized timers

• Disk throughput increase as network speeds increase
  – Disks can be organized in RAID0
Tolerating Overload Attacks Against Packet Capturing Systems

Antonis Papadogiannakis
Michalis Polychronakisis
Evangelos P. Markatos

papadog@ics.forth.gr
mikepo@cs.columbia.edu
markatos@ics.forth.gr