소프트웨어 기반 고성능 침입 탐지 시스템 설계 및 구현

*KyoungSoo Park*
Department of Electrical Engineering, KAIST

M. Asim Jamshed*, Jihyung Lee*, Sangwoo Moon*, Insu Yun*, Deokjin Kim‡, Sungryoul Lee‡, Yung Yi*
Department of Electrical Engineering, KAIST
‡ Cyber R&D Division, NSRI
Network Intrusion Detection Systems (NIDS)

- Detect known malicious activities
  - Port scans, SQL injections, buffer overflows, etc.
- Deep packet inspection
  - Detect malicious signatures (rules) in each packet
- Desirable features
  - High performance (> 10Gbps) with precision
  - Easy maintenance
    - Frequent ruleset updates
Hardware vs. Software

- H/W-based NIDS
  - Specialized hardware
    - ASIC, TCAM, etc.
  - High performance
  - Expensive
    - Annual servicing costs
  - Low flexibility
- S/W-based NIDS
  - Commodity machines
  - High flexibility
  - Low performance
    - DDoS/packet drops

IDS/IPS Sensors
- (10s of Gbps)
- ~ US$ 20,000 - 60,000

IDS/IPS M8000
- (10s of Gbps)
- ~ US$ 10,000 - 24,000

Open-source S/W
- ≤ ~2 Gbps
Goals

- High performance

• S/W-based NIDS
  - Commodity machines
  - High flexibility
Typical Signature-based NIDS Architecture

Packet Acquisition → Preprocessing → Multi-string Pattern Matching → Rule Options Evaluation

- Decode
- Flow management
- Reassembly
- Match Failure (Innocent Flow)
- Evaluation Failure (Innocent Flow)
- Evaluation Success
- Match Success

Rule Options Evaluation → Output
- Evaluation Success
- Malicious Flow
- Bottlenecks

* PCRE: Perl Compatible Regular Expression
Contributions

**Goal**
A highly-scalable software-based NIDS for high-speed network

**Slow software NIDS**
- Inefficient packet acquisition
- Expensive string & PCRE pattern matching

**Fast software NIDS**
- Multi-core packet acquisition
- Parallel processing & GPU offloading

**Bottlenecks**

**Solutions**

**Outcome**
- Fastest S/W signature-based IDS: 33 Gbps
- 100% malicious traffic: 10 Gbps
- Real network traffic: ~25 Gbps
Challenge 1: Packet Acquisition

- Default packet module: Packet CAPture (PCAP) library
  - Unsuitable for multi-core environment
  - Poor performance
  - More power consumption
- Multi-core packet capture library is required

Packet RX bandwidth: 0.4-6.7 Gbps
CPU utilization: 100%

* Intel Xeon X5680, 3.33 GHz, 12 MB L3 Cache
Solution: PacketShader I/O

- PacketShader I/O
  - Uniformly distributes packets based on flow info by RSS hashing
  - Source/destination IP addresses, port numbers, protocol-id
  - 1 core can read packets from RSS queues of multiple NICs
  - Reads packets in batches (32 ~ 4096)
- Symmetric Receive-Side Scaling (S-RSS)
  - Passes packets of 1 connection to the same queue

* S. Han et al., “PacketShader: a GPU-accelerated software router”, ACM SIGCOMM 2010
Challenge 2: Pattern Matching

- CPU intensive tasks for serial packet scanning

- Major bottlenecks
  - Multi-string matching (Aho-Corasick phase)
  - PCRE evaluation (if ‘pcre’ rule option exists in rule)

- On an Intel Xeon X5680, 3.33 GHz, 12 MB L3 Cache
  - Aho-Corasick analyzing bandwidth per core: 2.15 Gbps
  - PCRE analyzing bandwidth per core: 0.52 Gbps
Solution: GPU for Pattern Matching

- GPUs
  - Containing 100s of SIMD processors
    - 512 cores for NVIDIA GTX 580
  - Ideal for parallel data processing without branches
- DFA-based pattern matching on GPUs
  - Multi-string matching using Aho-Corasick algorithm
  - PCRE matching
- Pipelined execution in CPU/GPU
  - Concurrent copy and execution

Aho-Corasick bandwidth: 2.15 Gbps
PCRE bandwidth: 8.9 Gbps
Optimization 1: IDS Architecture

- How to best utilize the multi-core architecture?
- Pattern matching is the eventual bottleneck

<table>
<thead>
<tr>
<th>Function</th>
<th>Time %</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>acsmSearchSparseDFA_Full</td>
<td>51.56</td>
<td>multi-string matching</td>
</tr>
<tr>
<td>List_GetNextState</td>
<td>13.91</td>
<td>multi-string matching</td>
</tr>
<tr>
<td>mSearch</td>
<td>9.18</td>
<td>multi-string matching</td>
</tr>
<tr>
<td>in_chksum_tcp</td>
<td>2.63</td>
<td>preprocessing</td>
</tr>
</tbody>
</table>

* GNU gprof profiling results

- Run entire engine on each core
Solution: Single-process Multi-thread

- Runs multiple IDS engine threads & GPU dispatcher threads concurrently
  - Shared address space
  - Less GPU memory consumption
  - Higher GPU utilization & shorter service latency

GPU memory usage \(\frac{1}{6}\)
Architecture

- Non Uniform Memory Access (NUMA)-aware
- Core framework as deployed in dual hexa-core system
- Can be configured to various NUMA set-ups accordingly

▲ Kargus configuration on a dual NUMA hexanode machine having 4 NICs, and 2 GPUs
Optimization 2: GPU Usage

• Caveats
  – Long per-packet processing latency:
    • Buffering in GPU dispatcher
  – More power consumption
    • NVIDIA GTX 580: 512 cores

• Use:
  – CPU when ingress rate is low (idle GPU)
  – GPU when ingress rate is high
Solution: Dynamic Load Balancing

- Load balancing between CPU & GPU
  - Reads packets from NIC queues per cycle
  - Analyzes smaller # of packets at each cycle ($a < b < c$)
  - Increases analyzing rate if queue length increases
  - Activates GPU if queue length increases

Packet latency with
  - GPU: 640 μsecs
  - CPU: 13 μsecs
Optimization 3: Batched Processing

- Huge per-packet processing overhead
  - > 10 million packets per second for small-sized packets at 10 Gbps
  - reduces overall processing throughput
- Function call batching
  - Reads group of packets from RX queues at once
  - Pass the batch of packets to each function

\[ \text{Decode}(p) \rightarrow \text{Preprocess}(p) \rightarrow \text{Multistring\_match}(p) \]

\[ \text{Decode}(\text{list-p}) \rightarrow \text{Preprocess}(\text{list-p}) \rightarrow \text{Multistring\_match}(\text{list-p}) \]

2X faster processing rate
Kargus Specifications

**NUMA node 1**
- 12 GB DRAM (3GB x 4) - $100
- Intel 82599 Gigabit Ethernet Adapter (dual port) - $512
- NVIDIA GTX 580 GPU - $370

**NUMA node 2**
- Intel X5680 3.33 GHz (hexacore) 12 MB L3 NUMA-Shared Cache - $1,210

Total Cost (incl. serverboard) = ~$7,000
IDS Benchmarking Tool

• Generates packets at line rate (40 Gbps)
  – Random TCP packets (innocent)
  – Attack packets are generated by attack rule-set
• Support packet replay using PCAP files
• Useful for performance evaluation
Kargus Performance Evaluation

- Micro-benchmarks
  - Input traffic rate: 40 Gbps
  - Evaluate Kargus (~3,000 HTTP rules) against:
    - Kargus-CPU-only (12 engines)
    - Snort with PF_RING
    - MIDeA*
  
- Refer to the paper for more results

* G. Vasiliadis et al., “MIDeA: a multi-parallel intrusion detection architecture”, ACM CCS ’11
Innocent Traffic Performance

- 2.7-4.5x faster than Snort
- 1.9-4.3x faster than MIDeA

Actual payload analyzing bandwidth
Malicious Traffic Performance

• 5x faster than Snort
Real Network Traffic

- Three 10Gbps LTE backbone traces of a major ISP in Korea:
  - Time duration of each trace: 30 mins ~ 1 hour
  - TCP/IPv4 traffic:
    - 84 GB of PCAP traces
    - 109.3 million packets
    - 845K TCP sessions

- Total analyzing rate: 25.2 Gbps
  - Bottleneck: Flow Management (preprocessing)
Varying incoming traffic rates
- Packet size = 1518 B
Conclusion

• Software-based NIDS:
  – Based on commodity hardware
    • Competes with hardware-based counterparts
  – 5x faster than previous S/W-based NIDS
  – Power efficient
  – Cost effective

> 25 Gbps (real traffic)
> 33 Gbps (synthetic traffic)
US $~7,000/-
Thank You

fast-ids@list.ndsl.kaist.edu

https://shader.kaist.edu/kargus/