High-Speed Parallel Processing of Protocol-Aware Signatures

Jordi Ros-Giralt, James Ezick, Peter Szilagyi, Richard Lethin

Reservoir Labs
632 Broadway, #803
New York, NY 10012
(212) 780-0527
giralt@reservoir.com

Unclassified, DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited. This material is based upon works supported by the Department of Energy under contract numbers DE-FG02-08ER85046. Any opinions, findings and conclusions expressed in this material are those of Reservoir Labs, and do not necessarily reflect the views of the Department of Energy.
Copyright © 2009 Reservoir Labs, Inc.
Problem Definition and Previous Work

• **Why:** Intelligent signature matching and protocol parsing are key functions of intrusion detection systems that need to keep up with line rates (100 Gbps)
  – There exists a trade off between complexity of the signatures and processing speed.
  – Trend: cyber attacks become smarter, so need to handle complex signatures.
  – Trend: network trunks become faster, so need to handle high speed.

• **Problem:** How to process large number of protocol signatures at high speed?
  – Need to be able to process large number of signatures in parallel.
  – Need to minimize processing overhead on core processors.

• **How:** Use SAT tools, binary decision diagrams (BDD) and deterministic finite automatons (DFA)
  – Use SALT™ (converts Boolean functions into simplified Conjunctive Normal Form).
  – Compose OR signature, obtain its CNF form and calculate optimal BDD cuts.
  – Offload the resulting BDDs onto hardware accelerated DFA engines.

• **Result:** Capability to process large number of signatures at high speed onto hardware DFA engines
  – Currently building prototype on a heterogeneous 16-core processor/accelerator NPU (Cavium Octeon Plus) and looking at running this on the Octeon II and Tilera TILE64.
  – Experimental results presented in our abstract and poster.
How it Works Through a Low-Scale Example

Ghttpdlog signature expressed in salt:

```plaintext
;;;
;;; Ghttpdlog Salt
;;;

$bit1 expr =?
$bit2 expr =?
$bit3 expr =?
$bit4 expr =?
$bit5 expr =?
$conjunction1 and ~$bit1 $bit2 =
$conjunction2 and $bit1 ~$bit3
$bit4 =
$conjunction3 and $bit1 $bit3 $bit5 =
$disjunction or $conjunction1
$conjunction2 $conjunction3 =
$_ _ eval $disjunction + ; assert return value true

#done
```

Optimal min-max cut mapping on 3 DFA engines

HPEC 2009
22 September 2009
Results and Preliminary Conclusions

• **Average case analysis. Using a random signature generator and taking average cases:**
  - Emitting DFA specs for signatures from 6 to 20 variables and from 6 to 20 clauses per signature, the average DFA size obtained is **410340 bytes**.
  - That yields **2.43 signatures per megabyte**.
  - On a DFA with 256MB of memory, we can fit in average about **600 signatures**.
  - Each signature can involve 100s of CPU cycles per connection offloaded from the core processor.

• **Real examples. Examples using real signatures are shown in the poster.**

---

HPEC 2009
22 September 2009
Relationship with Previous Work

Schear’s Approach:
• A single signature only uses a small portion of the protocol state machine.
• By customizing the state machine to each signature (removing those elements in the state machine that are irrelevant to the signature), each signature can run much faster.

Our Approach:
• If N is large enough, then the union of Schear’s specialized state machines add up to the complete protocol state machine.
• In this case, it pays off to implement one single complete protocol state machine and have all signatures leverage the same machine.

Solutions in their own niches

Small number of signatures
- Sig 1
- Sig 2
- Sig N

Large number of signatures
- Complete protocol state machine
- S = Union of all signatures

HPEC 2009
22 September 2009