A Network Traffic Anomaly Detection Framework using GPUs

By
Meera Ramesh

October, 2016

Understanding Signature Analysis

- Signature – set of characteristics used to define a type of network activity
  - IP numbers and options, TCP flags, and port numbers are examples
- Some intrusion-detection devices assemble databases of “normal” traffic signatures
  - Deviations from normal signatures trigger an alarm
- Other devices refer to a database of well-known attack signatures
  - Traffic that matches stored signatures triggers an alarm

Examining Normal Network Traffic Signatures

- Important TCP flags
  - SYN (0x2) – *synchronize flag* is sent when a connection is initiated
  - ACK (0x10) – *acknowledgement flag* is set to signal that the previous packet was received
  - PSH (0x8) – *push flag* indicates that immediate delivery is required
  - URG (0x20) – *urgent flag* is used when urgent data is being sent
  - RST (0x4) – *reset flag* is sent when one computer wants to stop and restart the connection in response to a problem

Examining Normal Network Traffic Signatures

- Important TCP flags (cont'd)
  - FIN (0x1) – *finished flag* lets one computer know that the other is finished sending data

- Placement and use of these flags are definite
  - Deviations from normal use mean that the communication is suspicious

Understanding Signature Analysis

- Signature analysis:
  - Practice of analyzing and understanding TCP/IP communications to determine whether they are legitimate or suspicious
- Bad header information
  - Packets are often altered through header information
  - Suspicious signatures can include malformed
    - Source and destination IP address
    - Source and destination port number
    - IP options, protocol and checksums
    - IP fragmentation flags, offset, identification or checksum

Deep Packet Inspection

- Suspicious data payload
  - Payload
    - Actual data sent from an application on one computer to an application on another
  - Some IDPSs check for specific strings in the payload
- Suspicious data payload (cont'd)
  - Remote-access Trojans (RATs):open back doors that give the remote attacker administrative rights
  - Unix Sendmail program is exploited by adding codes to packet contents

Code Structure

- LibPcap a C++ library used for packet capture.
- Each packet was handled using the libpcap APIs and the captured information is saved in a struct datatype.
- The protocols analyzed were Ethernet, IPv4, IPv6, TCP, UDP and ICMP.
- Memory is allocated on the GPU for both the packets and the result. The packets are copied from CPU to GPU.
- Main modules
  - Auto Mining
  - Operation
  - Hooks

Thread Level Parallelism

- Each thread handles every packet.
- The kernel was launched with 30 blocks of 128 threads.
- Auto Mining
  - The data required for each thread is copied into shared memory, since it's faster.
  - Each packet has appropriate struct data type for TCP layer,etc. These struct types can be accessed by the GPU via the global memory.

- Operations
  - Each thread handles all the rules. If we identify the packet is malicious while checking the header, we discard the packet. If not, we go ahead to check the payload.

Thread Level Parallelism (cont'd)

- Operations (Cont'd)
  - Naive pattern matching algorithm was implemented
  - If any of the rule fails, the result will be updated. On return from the operation module. The result is copied from the GPU to the CPU and the hooks module is called.

- Hooks
  - Hooks module works on the CPU side.
  - The resultant values are checked to raise an alarm and drop the packets from further processing.

Block Level Parallelism

- Kernel is launched with 96 threads per block and 30 blocks.
- Rules are the same as thread Level Parallelism, but different approach. Here every block handles a single packet. Thus the threads in a block, each of them handle a byte of data.

- Auto Mining
  - Since each thread handles one byte, we need not copy the information of the attributes of every layer to each thread.

- Operation
  - The threads in the header region work on the header rules.
  - Warp Divergence handled efficiently.

Block Level Parallelism (cont'd)

- Operations (cont'd)
  - Threads in WARP 1 work on IP4 header rules, threads in WARP 2 work on TCP rules, threads in other 2 warps work on signature matching. To avoid threads being stalled due to if conditions.
  - Rabin Karp string matching algorithm was implemented for single patterns and multiple patterns.The patterns can be added automatically as new virus signatures are being found.

- Hooks
  - In the CPU side, each block result is checked. If found a malicious packet, alarm is raised.

Result

Offline capture mode was used to measure clock frequency.

- ThreadLevel Parallelism
- Clock Frequency = 18839.8us

- BlockLevel Parallelism
- Clock Frequency = 205us
  - With Rabin Karp algorithm

    Clock frequency = 112us

Future Scope

- Implement multiple string matching algorithms such as Boyer Moore, Aho-Corasick for Deep Packet Inspection and compare the performance.

Thank you for listening!