Motivations

- Ever-growing number of threats
- The market boost of embedded systems
- The increasing variety of operating systems
- Malware detection is a highly common and computationally-intensive problem in intrusion detection systems
What about small-scale systems?

- Great use of small-scale systems: mobile phones, gaming consoles, SoC etc.
- Memory and computation performance constraints
- Ever growth of attacks on small-scale systems
- Improvement in parallel computation performance

How to get benefit from parallel architectures to monitor small-scale systems?
Previous Work

• Development of parallel architecture for malware detection based on pattern matching technique
• Achieving better computing performance
• Use of Cuda and desktop GPU

Current Work

• Migration to mobile platform
• Use of OpenCL
• Building of behavioral malware dataset based on syscalls patterns
• Development of memory optimization techniques
Benefits of mobile GPGPU

✓ To ensure high security level of mobile devices, accelerating malware detection can be provided by GPU parallel processing

✓ Offering a complementary processing unit to the CPU

✓ Adapted to SIMD architecture

✓ Fast memory types access (shared memory, constant memory)

✓ More and more evolving

✓ GPUs driven by high-end applications; prepared to pay a premium for high performance
Evolution of GPUs for embedded systems
**Popular mobile GPUs**

**Adreno 330**
- Inbuilt in Snapdragon™ 800 Series Processors.
- Speed can push to 3.6 gig pixels/s
- Used in HTC one, Xperia z ultra

**Power VR SGX544mp3**
- Inbuilt in Exynos 5 Octa processor
- Used in galaxy s4 @ 533 MHz clock speed

**Mali T604**
- First time used in Exynos 5
- The 1st Midgard architecture gpu for arm
- Used in famous series of Google tablet nexus 10.
GPU architecture

Parallel Processing architecture
Architecture Challenges

1. Which type of dataset can we use in order to have good detection accuracy?

2. How can we increase GPU processing performance?

3. The need of applying memory compacting techniques

Limitation of Mobile GPU memory VS important memory requirement of DFA
Which type of dataset can we use in order to have good detection accuracy?
Malware detection

Extraction of malicious behavior based on syscalls sequences with the thread-grained extraction technique

- Key concept: Malwares which have the same malicious code embedded on benign applications, will have common malicious behaviour
- Tracking the tree architecture of applications threads
- Malicious behaviour is likely to appear at the same thread level on applications having the same malware

Thread tree architecture of two applications belonging to the same malware family
Malware detection

Training phase

**Recording Phase**
- Having different applications that belong to the same malware family
- Execution of every application
- Tracking thread tree structure of the application
- Recording syscalls sequences for every thread created by the application

**Extraction phase**
- Extraction of common syscalls subsequences belonging to threads from the same family and having the same level
- Storage of common subsequences
Malware detection

Training phase

Filtering Phase

- Get syscall patterns from benign applications B
- For every Csi in our malware dataset M
  - Counting the number of common subsequence Csi appearing in B and in M
  - Calculate malicious probability of Csi
  - Storing Csi into Malware common subsequences if Csi has high malicious probability
- We choose to work with Csi having malicious probability = 1 (Csi appearing in M and not in B)

The result of training phase = Malware behavioural dataset build of syscall patterns
(1) Which type of dataset can we use?

(2) How can we increase GPU processing performance?
Challenge 2

How can we increase the parallel processing performance of pattern matching on the GPU?
Parallel Pattern Matching Algorithm

- Match of data streams by malware scanner against a large set of known signatures, using a pattern matching algorithm.
- Pattern matching algorithms analyze the data stream and compare it against a database of signatures to detect known malware.
- Fairly complex signature patterns composed of different-size strings, range constraints, and sometimes recursive forms.

Example
- Aho-corasick
- Wu-manber
- Knuth-Morris-Pratt
Aho-Corasick

- AC algorithm is based on a DFA structure built from reference patterns.
- The construction of automaton is done in pre-processing phase.
- The matching process is done in processing phase.
- The automaton structure can be essentially described by two tables: transition table and failure state table.
Direct implementation of parallel pattern matching

- **Idea**
  - Input stream segmentation
  - For every segment we associate a thread
  - Problem of boundary detection

- **Possible solution**
  - Every thread check the pattern presence on the edges.
  - Each thread must scan for a minimum length which is $s$ almost equal to the segment length plus the longest pattern length of an AC state machine.

![Diagram](image-url)
Parallel Pattern Matching Algorithm

Aho-Corasick


• **Goals**
  - Increase pattern matching computation throughput via parallelization.
  - Resolve the throughput bottleneck caused by the overlapped computation.

• **Idea**
  - Byte allocation per thread
  - Failure transitions elimination
  - The thread stops his work if no valid transition is found.
Parallel Failureless Aho-Corasick
Parallel Pattern Matching Algorithm

Parallel Failureless Aho-Corasick

- Increase of the algorithm performance on GPU
Challenge 3

• Malwares grows continuously

• The number of signatures is increasing proportionally

➤ Scaling problems for mobile anti-malwares due to:

• Limitation of Mobile GPU memory VS Important memory requirement for DFA structure

The need of applying memory compacting techniques
Eliminating failure transition

Eliminating Final states table by performing state reordering

Applying P3FSM technique

Memory optimization technique

1

2

3
P3FSM: Portable Predictive Pattern Matching Finite State Machine

(a) DFA for Patterns “SHE”, “HERS”, “HIS”

Memory optimization technique

<table>
<thead>
<tr>
<th>Code Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Character/Cluster Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Char</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>H</td>
</tr>
<tr>
<td>S</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

1. Check cluster $\triangle$ of state code 7 for character signature of $H \square 00 = 00$
2. Compute next state: state signature + character offset $\square 00 + \square 00 = 2 ->$ next state index $= \square 2$, next state $= \square 5$
Experimentations

- **Hardware**
  - Mobile Phone
    - HTC one
  - GPU
    - Adreno 320
  - CPU
    - Qualcomm Snapdragon 600, quad-core CPU @ 1.7GHz

- **Benchmark**
  - 600 Malicious syscalls patterns
Experimentations

Thread per block resizing

Local work-group resizing

✓ Best throughput with 64x64 threads = 72Mb/s
Effective use of the different GPU memory types

- Around 15% gain of performance with the use of constant and shared memories
- Applying shared memory to improve the latency of global memory accesses
An acceleration of around 2.3x is obtained with the parallel processing on the mobile GPU over serial processing.

The framework throughput is dominated by data transfers between the host/device which consist of 60% of the total processing time.
Storing DFA structure on the GPU is memory consuming especially that mobile GPU memory is small.

Difference in memory requirement between PFAC DFA and P3FSM. 

P3FSM that compacts the DFA structure by many times comparing to standard PFAC DFA.
Conclusion

- Implementation of a parallel host-based anti-malware on mobile GPU using behavioral detection techniques

- Series of optimizations to deal with the low memory problem of mobile devices and the ever-increasing computing and memory requirements of malware detection

**Perspectives:**
- Integrating a GPU monitor which tracks down the GPU memory usage and allows the automaton adjustment in real-time to fit the reduced GPU memory
- Use of mobile GPU clusters
- Working on malware dataset to achieve better detection accuracy
Thank you for your attention