PixelVault: Using GPUs for Securing Cryptographic Operations

Pengfei Sun
CS516: Programming Language and Compilers II
Rutgers University
1. Motivation

2. Related Work

3. Solution

4. Results
1. Motivation
• Secret keys **may remain unencrypted** in CPU Registers, RAM, etc.
  – Memory attacks
  – DMA attacks
  – **Heartbleed** attack

Authentication, Confidentiality, Integrity
Heartbeat – Normal usage

Client

Server, send me this 4 letter word if you are there: "bird"

bird

Heartbeat – Malicious usage

Client

Server, send me this 500 letter word if you are there: "bird"

bird. Server master key is 31431498531054. User Carol wants to change password to "password 123"...
2. Related Work
If the operating system kernel is exploited, all security guarantees are null and void.
CPU Solutions

• Virtualization
  – Overshadow
  – TrustVisor (a piece code)
  – InkTag
  – Qubes-OS
  – AppGuard

• Compiler
  – VirtualGhost
GPU

• Protection by using GPUs
  – Implemented encryption algorithm

• Security Attacks using GPUs
  – GPU-based keylogger
  – GPU-based cracking passwords

• Malicious Attacks on GPUs
  – Stealing webpages rendered on your browser by exploiting gpu vulnerabilities
  – Cuda leaks: information leakage in gpu architectures
  – Vulnerability analysis of GPU computing
Simple Example?

```c
#include <stdio.h>
#include <cuda.h>

int main()
{
    float *a_device;
    float a_host = 80.0;
    cudaMalloc((void **)&a_device, sizeof(float));
    cudaMemcpy(a_device, &a_host, sizeof(float), cudaMemcpyHostToDevice);
    printf("The value %f was written to the device.\n", a_host);
    cudaFree(a_device);
    return 0;
}
```

```
#include <stdio.h>
#include <cuda.h>

int main()
{
    float *a_device;
    float a_host;
    cudaMalloc((void **)&a_device, sizeof(float));
    cudaMemcpy(&a_host, a_device, sizeof(float), cudaMemcpyDeviceToHost);
    printf("The value %f was written to the device.\n", a_host);
    cudaFree(a_device);
    return 0;
}
```
3. Solution
PixelVault Overview

- Runs encryption securely outside CPU/RAM
- Only on-chip memory of GPU is used as storage
- Secret keys are never observed from host
Challenges

• How to italicize GPU execution?

• Who bolds the keys?

• Where is the code?
How to isolate GPU execution

- Handled by the host
  - Load parameters, launch GPU kernel, transfer data, etc.

- Not secure for our purposes
  - Crypto keys have to be transferred every time

- Force GPU kernel to run indefinitely
  - Using an infinite while loop

- Cannot rely on the typical parameter-passing execution of GPU kernels
  - Allocated a memory segment that is shared between GPU/CPU
Shared Memory between CPU/GPU

- Page-locked memory
  - Accessed by the GPU directly, via DMA
- Processing requests are issued through this shared memory space
- GPU continuously monitors the shared space for new requests
Autonomous GPU execution

- Non-preemptive execution
- Only the output block is being written back to host memory
Who holds the keys?

Off-chip global memory. No protection; data can be acquired by the CPU directly.

The data cannot be managed by the programmer.

Reset to zero on each GPU kernel execution.
Keep secrets on GPU registers

• Secret keys are loaded on GPU registers at an early stage of the bootstrapping phase
  – Preferably from an external storage device

• Unfortunately, the number of available registers in current GPU models is small
  – Enough for a single/few secret keys, but what about multi-homing servers?
Support for an arbitrary number of keys

- We can use a separate KeyStore array that holds an arbitrary number of secret keys.
• GPU code is initially stored in global device memory for the GPU to execute it
  – An adversary could replace it with a malicious version

Where is the code?

• Solution: load the code to icache, and then erase it from global device memory
  – The code runs indefinitely from the icache
  – Not possible to be flushed or modified
4. Results
AES-128 CBC Performance (#Msgs/sec)

3x-4x faster than CPU for a sufficient number of messages
RSA 1024-bit Performance (#Msgs/sec)

<table>
<thead>
<tr>
<th>#Msgs</th>
<th>CPU</th>
<th>GPU [25]</th>
<th>PixelVault</th>
<th>PixelVault (w/ KeyStore)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1632.7</td>
<td>15.5</td>
<td>15.3</td>
<td>14.3</td>
</tr>
<tr>
<td>16</td>
<td>1632.7</td>
<td>242.2</td>
<td>240.4</td>
<td>239.2</td>
</tr>
<tr>
<td>64</td>
<td>1632.7</td>
<td>954.9</td>
<td>949.9</td>
<td>939.6</td>
</tr>
<tr>
<td>112</td>
<td>1632.7</td>
<td>1659.5</td>
<td>1652.4</td>
<td>1630.3</td>
</tr>
<tr>
<td>128</td>
<td>1632.7</td>
<td>1892.3</td>
<td>1888.3</td>
<td>1861.7</td>
</tr>
<tr>
<td>1024</td>
<td>1632.7</td>
<td>10643.2</td>
<td>10640.8</td>
<td>9793.1</td>
</tr>
<tr>
<td>4096</td>
<td>1632.7</td>
<td>17623.5</td>
<td>17618.3</td>
<td>14998.8</td>
</tr>
<tr>
<td>8192</td>
<td>1632.7</td>
<td>24904.2</td>
<td>24896.1</td>
<td>21654.4</td>
</tr>
</tbody>
</table>
• Cryptography on the GPU is not only fast
• Also secure

Conclusion

– Preserves the secrecy of keys even when the base system is fully compromised
Q&A