Accelerating auditing of slow hashes by using GPUs.

Dhiru Kholia (dkholia@cs.ubc.ca)

January 11, 2011

1 Proposal Summary

Password auditing is a frequently used technique to find out weak user passwords. It is a popular and useful exercise for the following reasons:

- The presence of weak and default passwords represents significant gaps in the security policy of an organization. Proactive auditing of user credentials to make sure that only good passwords are used is important to reduce the size of the password cracking attack vector.
- RockYou incident revealed that a significant number of users use trivial / simple and easy to remember passwords. Proactive auditing and banning of such trivial / dictionary based passwords can help protect against online passwords attacks [6].

I plan to extend John the Ripper [7] and Alain Espinosa's OpenCL patches for cracking NTLM hashes on GPU by adding support for cracking slow hashes like FreeBSD MD5 and Blowfish using GPU(s). I believe that this project is important for the following reasons:

- Password auditing is an embarrassingly parallel problem but it is a computationally intensive process. Therefore, making it a good candidate for parallelization.
- GPUs computing is at its tipping point and has already shown impressive speedups in diverse fields [5].
- While DES, LANMAN and NTLM hashes can be cracked at high speeds on CPUs [8], the problem of cracking slower hashes like FreeBSD MD5 and Blowfish at high speeds remains unsolved. The "slower" hashes employ techniques like salting and key stretching, thus making them more computationally intensive than normal hashes.
- There are commercial companies which already utilize GPU(s) for password auditing (Elcomsoft). However, such proprietary and closed-source services / solutions are expensive and don't contribute to research in my opinion.

My work will involve extending John the Ripper 1.7.6 and Alain Espinosa's OpenCL patches in order to crack slow hashes like FreeBSD MD5 and Blowfish. A major (and ambitious) aim of this project is to merge my work upstream. I also plan to make extensive use of PyOpenCL framework for prototyping. I will consider my project to be successful if the following goals (or a significant subset of the mentioned goals) are met:

- Obtain a speedup of at least 5X on ATI Radeon™ HD 4870 GPU compared to single-core performance on an Intel(R) Core(TM) i5 750 CPU for **"some"** hashes.

- Upstream accepts and merges my patches into the John the Ripper's jumbo patchset.
- Understand the challenges involved in programming GPUs. In particular, understand the various performance tweaks needed to utilize GPUs efficiently.
- Understand current (and potential future) bottlenecks which limit the performance of GPUs for auditing passwords.

2 Overview of my project work

I started my project work by modifying examples included with PyOpenCL (implemented trivial kernels to do squaring / addition of input data). After reading OpenCL guides and tutorials published by nVIDIA and AMD I started implementing OpenCL programs in pure C. By around 3rd week of December I had accumulated enough background knowledge (and courage) to actually study the OpenCL patch published by Alain. After some hacking sessions, I started posting to john-users mailing list (the whole development history of my work can be found at http://thread.gmane.org/gmane.comp.security.openwall.john.user/3067). My specific extensions to Alain's OpenCL patch are:

- Fix compilation of Alains's patches on ATI cards (AMD's OpenCL implementation seems to be a bit more strict / restricted than nVidia in not allowing goto keyword).
- Rebase Alain's patches on jumbo-9 patch instead of jumbo-7 patch.
- Implement MD4 kernel and extend JtR to enable cracking of raw-MD4 hashes.
- Implement MD5 kernel and extend JtR to enable cracking of raw-MD5 hashes.
- Minor tweaks to enable the printing of compilation failure logs (useful for debugging).

I did not implement OpenCL support for Blowfish and FreeeBSD MD5 mainly because there were other "easier" targets which sufficed as my project goals. Even with GPU acceleration I don't think that attacking these hashes is practically feasible for now (weak justification for being lazy!).

The most surprising outcome of this project was see sub-$100 GeForce GT 240 outperforming $150 ATI 4870 and $400 ATI 6970 cards. The low OpenCL performance of ATI 48XX might be explained in part by the fact that local memory is emulated in global memory due to hardware limitations. However, it is hard to pin-point the cause of relative low performance on ATI 69XX cards (Alain hints at looking into memory usage and vectorization guidelines published by AMD). I blame it on AMD's OpenCL compiler (nVIDIA's CUDA has been in market for a while now) and missing support for 69XX series in Catalyst 10.12 video driver. On a more serious note, I don't see the whole point of OpenCL if I have to resort to write vectorized code by hand (CAL) limited to a particular manufacturer. So for now, achieving high performance on AMD GPUs remains as a future work. I correctly predicted that this project would take much more time than 40 hours (GPU programming, high-speed cryptography are complex subjects and I have no experience with either of them) in my initial proposal. Initially, I thought that learning GPU and some cryptography would take the majority of the time. However, it turned out that I could re-use lot of existing code and most of my time was spent in debugging OpenCL kernels!

Overall, the 4X to 6X (so far!) performance improvements achieved on a sub-$100 GeForce GT 240 are very promising. It would be interesting to see how these OpenCL patches perform on high-end nVIDIA GPUs ($500 ones!) as a future exercise. Sections 3 and 4 demonstrate the performance improvements made by using my OpenCL patches.

3 Performance on CPU (Intel Core i5 750 @ 2.67GHz)

```
[dsk@anka ~]$ tar -xjf john-1.7.6.tar.bz2
[dsk@anka ~]$ cd john-1.7.6/
[dsk@anka john-1.7.6]$ zcat ../john-1.7.6-jumbo-9.diff.gz | patch -p1
...
[dsk@anka john-1.7.6]$ cd src/
[dsk@anka src]$ make linux-x86-64 -j8
...
[dsk@anka src]$ ../run/john -test -format=raw-md4
Benchmarking: Raw MD4 [32/64]... DONE
Raw: 6136K c/s real, 6198K c/s virtual

[dsk@anka src]$ ../run/john -test -format=raw-md5
Benchmarking: Raw MD5 [raw-md5 64x1]... DONE
Raw: 5836K c/s real, 5895K c/s virtual

[dsk@anka src]$ ### Meh ... ###
```

4 Performance on nVIDIA GPU (nVIDIA GeForce GT 240):

```
[dsk@anka ~]$ tar -xjf john-1.7.6.tar.bz2
[dsk@anka ~]$ cd john-1.7.6/
[dsk@anka john-1.7.6]$ zcat ../john-1.7.6-jumbo-9.diff.gz | patch -p1
...
[dsk@anka john-1.7.6]$ ### applying OpenCL patches ###
[dsk@anka john-1.7.6]$ zcat ../john-1.7.6-jumbo-9-opencl-9.diff.gz | patch -p1
...
[dsk@anka john-1.7.6]$ cd src/
[dsk@anka src]$ make linux-x86-64-opencl -j8
...
[dsk@anka src]$ ../run/john -test -format=raw-md4
Benchmarking: Raw MD4 [32/64]...
OpenCL Platform: <<<NVIDIA CUDA>>> and device: <<<GeForce GT 240>>>
DONE
Raw: 27698K c/s real, 61166K c/s virtual

[dsk@anka src]$ ../run/john -test -format=raw-md5-opencl
Benchmarking: Raw MD5 [raw-md5-opencl]...
OpenCL Platform: <<<NVIDIA CUDA>>> and device: <<<GeForce GT 240>>>
DONE
Raw: 37008K c/s real, 111025K c/s virtual

[dsk@anka src]$ ### Hazzah :-) ###
```

5 Performance on ATI GPU (ATI 6970, top of the line card):

```
[dsk@godbox ~]$ tar -xjf john-1.7.6.tar.bz2
[dsk@godbox ~]$ cd john-1.7.6/
[dsk@godbox john-1.7.6]$ zcat ../john-1.7.6-jumbo-9.diff.gz | patch -p1
...
[dsk@godbox john-1.7.6]$ ### applying OpenCL patches ###
[dsk@godbox john-1.7.6]$ zcat ../john-1.7.6-jumbo-9-opencl-9.diff.gz | patch -p1
...
[dsk@godbox john-1.7.6]$ cd src/
[dsk@godbox src]$ make linux-x86-64-opencl -j8
...
[dsk@godbox src]$ ../run/john -test --format=raw-md4
Benchmarking: Raw MD4 [32/64]...
OpenCL Platform: <<<ATI Stream>>> and device: <<<Cayman>>>
DONE
Raw: 31956K c/s real, 32577K c/s virtual

[dsk@godbox src]$ ../run/john -test --format=raw-md5-opencl
Benchmarking: Raw MD5 [raw-md5-opencl]...
OpenCL Platform: <<<ATI Stream>>> and device: <<<Cayman>>>
DONE
Raw: 33554K c/s real, 33893K c/s virtual

[dsk@godbox src]$ ### Hmmm ... FIXME! ###
```

6 Future Work, Limitations and Open Questions:

- Implement support for cracking FreeBSD MD5 and Blowfish hashes on GPU using OpenCL. Since cracking these hashes on CPU is very slow, the relative performance gains should be very good. However implementing Blowfish on a GPU might be challenging due to its large internal state [13].
- My current approach involves generating keys on the CPU and then transferring to the GPU in chunks (also hashes are transferred back to the CPU in a somewhat optimal fashion). This causes stalls on both CPU and the GPU leading to sub-optimal performance. This approach is forced by the current architecture of JtR which is not threaded. It seems practical improvements can be made by using multiple command queues and extending JtR to utilize multiple CPU cores. However, this option wasn't investigated due to lack of time and complexity involved.
- Is it possible to implement a complex key generation algorithm (like JtR) on a GPU or will brute-force key generation (simpler to implement) suffice? If key generation is done on the GPU, then lot of architecture (single CPU will suffice) and performance issues (like PCI Express bottlenecks) simply disappear.
- whitepixel [12]achieves a speed of 4 billion passwords on AMD 6970 GPU. However it uses AMD CAL IL and not OpenCL. It is possible to get similar speed ups using OpenCL and C? What kind of changes will it involve? (Currently whitepixel is limited to cracking only single-hash). *This would be the follow up project which I would do given another opportunity.*

- Implement and study the effect of dynamic key sizes on performance empirically. While trivial to implement and test, this could not be completed due to lack of time. This explains the lack of fancy graphs in this report.
- Implement temperature monitoring for GPUs. Can this be done in a device independent manner (probably not)? At some point this feature needs to be implemented in order to safeguard the hardware. Look into AMD GPUPerfAPI (for measuring GPU utilization) and AMD ADL (for measuring temperature).

7 Conclusions and Rants:

Programming GPU and getting decent performance boost is not a trivial task; it is an exercise filled with frustration with periods of great elation (and relief) primarily due to amount of debugging involved (at least in my case). I imagine that GPU development will become easier (debugging and profiling tools will catch up making life easier for OpenCL developers) in near future. Delaying writing project reports towards the end of the project only promotes further procrastination. This fact was re-discovered during this project!

Overall, I think this project is my most successful project so far in terms of its usefulness in real life applications, application of my research skills into a whole new area and in terms of the visibility / attention received upstream.

8 Acknowledgments:

- Solar Designer (author of John The Ripper) : for writing the best password cracking program and making it open-source!
- Alain Espinosa (author of OpenCL patch) : my project couldn't have possibly started without Alain's excellent work.
- Erik Winkler : for running my experimental (and buggy patches) and providing useful feedback and benchmarks. Thank you!
- Users of john-user mailing list : thanks for testing the patch and providing feedback!

References

[1] Schneier, B. 1994. Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish). In Fast Software Encryption, Cambridge Security Workshop (December 09 - 11, 1993). R. J. Anderson, Ed. Lecture Notes In Computer Science, vol. 809. Springer-Verlag, London, 191-204.

[2] Rivest, R. 1992 The MD5 Message-Digest Algorithm. RFC. RFC Editor.

[3] Song Jun Park; Shires, D.R.; Henz, B.J.; , "Coprocessor Computing with FPGA and GPU," DoD HPCMP Users Group Conference, 2008. DOD HPCMP UGC , vol., no., pp.366-370, 14-17 July 2008

[4] Sangjin Han; Keon Jang; KyoungSoo Park; Moon, S.; , "Building a single-box 100 Gbps software router," Local and Metropolitan Area Networks (LANMAN), 2010 17th IEEE Workshop on , vol., no., pp.1-4, 5-7 May 2010

[5] John Nickolls, William J. Dally, "The GPU Computing Era," IEEE Micro, pp. 56-69, March/April, 2010

[6] Analysis of leaked RockYou's password database (containing 32 million credentials) by Matt Weir, http://reusablesec.blogspot.com/2009/12/rockyou-32-million-password-list-top.html

[7] John the Ripper, http://www.openwall.com/john/

[8] John the Ripper benchmarks, http://openwall.info/wiki/john/benchmarks

[9] ElcomSoft Distributed Password Recovery, http://www.elcomsoft.com/edpr.html

[10] Alain Espinosa's OpenCL patches for cracking NTLM hashes, http://www.openwall.com/lists/john-users/2010/09/06/2

[11] OpenCL overview, http://en.wikipedia.org/wiki/OpenCL

[12] whitepixel : fastest MD5 hash cracker, http://whitepixel.zorinaq.com/

[13] Ivan Golubev's blog : http://www.golubev.com/blog/?p=94