Accelerating auditing of slow hashes by using GPUs.

Dhiru Kholia (dkholia@cs.ubc.ca)

January 11, 2011

1 Proposal

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 by adding support for cracking slow hashes like FreeBSgD 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. However, such proprietary and closed-source services / solutions are expensive and don't contribute to research (in my opinion).

2 Plan :

I will be reading and posting to "john-users" mailing list (http://www.openwall.com/lists/john-users/). In addition, I will study the John the Ripper 1.7.6 source code and Alain Espinosa's OpenCL patches for cracking NTLM hashes [10]. The list of references which I plan to study is mentioned below.

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

2.1 Software List

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 project work upstream. I also plan to make extensive use of PyOpenCL framework for prototyping.

3 Assessment

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.

4 Notes

This project will definitely take much more time than 40 hours (GPU programming, high-speed cryptography are complex subjects and I have no experience with either of them).