Google Summer of Code

REPORT

Qualification Task

Katja Malvoni
Mentoring organization: *Openwall*

May 2013.

Task description

The task was implementation of the bcrypt hashing algorithm on the Epiphany chip. My approach was to get to know the chip and understand bcrypt algorithm before working on the task itself. To do that I went trough Epiphany Architecture Reference, Epiphany SDK Reference and read the paper A Future-Adaptable Password Scheme by Niels Provos and David Mazières. I decided to reuse existing Openwall bcrypt implementation because of multiple reasons. Epiphany SDK supports C, there is possibility that implementing bcrypt from scratch would result with bugs and it is highly unlikely that form scratch implementation could outperform existing one.
My first approach was to load whole bcrypt on a single Epiphany core. I started with only one core because the idea is to have one bcrypt instance per core so all cores will execute the same code. And I found it simpler to make it work on one core only because I never used Epiphany before. The first problem I encountered was the code size - the whole code couldn't fit core internal memory. I first used legacy.ldf instead of fast.ldf to have all the code in the external SDRAM. After that, I compiled code using compiler optimization options and the code could fit the internal SRAM (fast.ldf used). After my posting to the john-dev mailing list, Alexander suggested using size-optimized bcrypt implementation: `http://git.musl-libc.org/cgit/musl/tree/src/crypt/crypt_blowfish.c`. This implementation fits the internal SDRAM without any compiler optimizations and it is used in the attached code (e_bcrypt.c). It was tested with only one test vector and it produced the correct result. I wanted to test it with more vectors, do self-test and to measure performance but the system crashed and I wasn't able to do the tests nor measure the performance. I also wanted to test the performance of three different working .elf files (legacy.ldf, compiler optimizations and size-optimized) and compare them. I was mostly interested in performance difference between having code in the internal SRAM versus code being placed in the external SDRAM.

Encountered problems

Except the problem with code size, I encountered problems related to the system itself. When I first started working on the qualification task, one version of the eSDK was installed. After I reported problems with compilation, Yaniv updated the system. What happened to me is that I had been using outdated matrix multiplication example as my reference and I lost some hours trying to use same library as in outdated example. The

library has a different name in the newest release and the functions I had been trying to use also changed. The lesson which I learned from this problem is that I should ask questions even though they sound basic and unimportant.

Future work

Future work is to run bcrypt on all the cores, not just a single core. Except, that, Alexander suggested implementing only the most costly loop on the Epiphany. The two implementations should be compared and the better one should be integrated into John the Ripper.