Backtrack 4

CUDA GUIDE

by

Pureh@te

Version 2
# Table of Contents

What is CUDA? ............................................................................................................................................... 3
Supported GPUs ............................................................................................................................................. 3
Why do I care about CUDA? .......................................................................................................................... 3
Where can I get this CUDA thing? .................................................................................................................. 4
What is CUDA not? ......................................................................................................................................... 4
Getting started .............................................................................................................................................. 5
Nvidia-DIvers. ................................................................................................................................................ 5
Overclocking ................................................................................................................................................ 6
Installing the CUDA toolkit and SDK .............................................................................................................. 9
CUDA-Multiforcer ......................................................................................................................................... 13
Crack........................................................................................................................................................... 15
Pyrit ............................................................................................................................................................. 17
What is pyrit? .............................................................................................................................................. 17
Up and running with pyrit ............................................................................................................................... 18
Making sure Pyrit is working .......................................................................................................................... 19
List cores..................................................................................................................................................... 19
Benchmark.................................................................................................................................................. 20
WPA Wordlist............................................................................................................................................... 21
Attack_passthrough Mode............................................................................................................................... 22
Attack_passthrough with Crunch.................................................................................................................. 23
Batch............................................................................................................................................................. 24
Analyze........................................................................................................................................................ 26
Strip............................................................................................................................................................... 27
Building aircrack-ng with CUDA support: .................................................................................................... 28
Useful Links: ............................................................................................................................................... 30
What is CUDA?

CUDA (an acronym for Compute Unified Device Architecture) is a parallel computing architecture developed by NVIDIA. CUDA lets programmers utilize a dedicated driver written using C language subroutines to offload data processing to the graphics processing hardware found on Nvidia's latemodel GeForce graphics hardware. The software lets programmers use the cards to process data other than just graphics, without having to learn OpenGL or how to talk with the card specifically. Since CUDA tools first emerged in late 2006, Nvidia’s seen them used in everything from consumer software to industrial products, and the applications are limitless.

Supported GPUs

A complete list of supported GPU's can be found at the following link:

http://en.wikipedia.org/wiki/CUDA#Supported_GPUs

Why do I care about CUDA?

Hardware acceleration of password recovery is possible with CUDA enabled applications. Many of these applications are already available and there are many more to come. The support of NVIDIA graphic accelerators increases the recovery speed by an average of 10 to 15 times faster than was previously possible.
Where can I get this CUDA thing?

Backtrack 4 final comes fully ready to execute and build CUDA powered applications. I will review some of the major points involved in setting up the environment and running some of the application.

What is CUDA not?

CUDA is not a magic bullet that will suddenly make all software on an Nvidia-equipped PC run dramatically faster, in other words -- the programmer needs to figure out where the program can be optimized to process data in parallel. But within that context, programming support for CUDA can make a big difference.
Getting started

Nvidia-drivers:

The first thing we need to do is get the nvidia drivers installed. This is done easily with Backtracks package manager apt-get or aptitude. Installing the nvidia drivers is best done while the X server is not running. The X server can be stopped by pressing ctrl - alt -backspace.

```
root@bt:~# aptitude search nvidia-driver
p  nvidia-driver                  - Driver for nvidia graphics cards 2.6.30.9
root@bt:~# aptitude install nvidia-driver
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading extended state information
Initializing package states... Done
The following NEW packages will be installed:
  nvidia-driver
0 packages upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 11.4MB of archives. After unpacking 81.4MB will be used.
Writing extended state information... Done
0% [Waiting for headers1]
```

Once you get the drivers installed, a new xorg-config should be generated for you and then you can “startx” and return to the kde desktop environment.

You can make sure the driver was properly installed by greping the xorg.conf for the term nvidia.
In the event the auto xorg.conf does not work, nvidia provides a utility which may be able to help. To invoke it simply type “nvidia-xconfig” into a terminal and it will try to generate a new xorg config for you.

If you have multiple monitors you may need to use the nvidia-settings tool to configure them. In order to use the settings tool, either launch it from the KDE menu or run the command “nvidiasettings” in a terminal. The actual configuration is beyond the scope of this document however it is fairly easy to understand.

**Overclocking:**

There are two ways to overclock your video card in Linux. The first way is to use the nvidiasettings tool which comes with the nvidia-driver. In order to do this you need to edit your xorg.conf in order to unlock the option.

```
nano /etc/X11/xorg.conf
```

and find the section that looks like this:
Section "Device"

Identifier "Videocard1"
Driver "nvidia"

VendorName "NVIDIA Corporation"
BoardName "GeForce 8800 GT"
BusID "PCI:3:0:0"

Screen 1

Option "AddARGBGLXVisuals" "true"

Option "Coolbits" "1"

Option "RenderAccel" "true"

EndSection

Add the coolbits option and then restart X and open nvidia-settings and you should have a overclock option like this:
The second way to overclock your card in Linux is to use the nvclock command line utility. As you can see there are many options with the nvclock utility. Please ensure your settings before overclocking your card in this manner. I normally use the gamer forums as a good source for finding out optimal clock settings for various cards.
Installing the CUDA toolkit and SDK:

Now that we have our driver installed and the clock settings to our liking, it is time to get our CUDA development environment set up. This is not necessary if you are only interested in running a tool such as Pyrit however if you want to build any CUDA applications you will need this environment.

The environment is already built and set up so we simply need to apt-get it. This will require about 250 MB of space so make sure you have the space to set this up. The tool kit should be installed first. This contains all of our tools for working with CUDA applications such as the nvcc compiler and the cuda debugger.

The next package is the CUDA-SDK and contains coda samples and some of the default make files we can use to build CUDA applications.
The CUDA compilers and the libraries should be added to your path by the package manager. The environment variables are, however, set up for root so if you have created another user you will need to move the following piece of root's .bashrc to your new user.

Once you have installed the packages you need to update your $PATH variable and make sure the CUDA compiler is in the path.
If your output looks like mine pictured here then you are good to go.

The code sample projects are located in the `/opt/cuda/sdk/C/src/` directory if you want to play around with them.

Basically you go into the directory of the tool you want to build and issue the make command. The nvcc compiler then builds the tool and outputs the binary into the `/opt/cuda/sdk/C/bin/linux/release` directory and then can be used.
CUDA-Multiforcer:

One of the newest tools in Backtrack 4 is the CUDA-Multiforcer. This is a password bruteforcer which supports MD4 / MD5/ NTLM and now SHA1 hash’s. It is incredibly fast and can greatly decrease the time it takes to crack password hash’s while on a pentest. Installation of the multi-forcer is simple.
Running the multi-forcer with no arguments will show you the required options for running the tool.

root@bt:/pentest/passwords/cuda-multiforcer# ./CUDA-Multiforcer

Currently supported hash types: MD5 FASTMD5 MD4 FASTMD4 NTLM FASTNTLM SHA1 FASTSHA1

./CUDA-Multiforcer: missing option -h|--hashtype={NTLM, MD4, MD5}

./CUDA-Multiforcer: missing option -f|--hashfile=hashfile

./CUDA-Multiforcer: missing option --min=<n>

./CUDA-Multiforcer: missing option --max=<n>

Here is an example of running the Cuda-multi-forcer against a NTLM hash
There are different charset files in the charsets directory or you can make a custom one. The syntax of the file is very simple.

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ\[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

The –f argument requires a single hash or a file of hash’s

The multi-forcer can also handle very large lists of hash’s. I loaded 4000 MD5 passwords as a test and the tool easily loaded them all and started cracking.

The only limitation of this tool I can find is that it only supports one GPU card. This example was tested on a 8800 GT card but I have run the tool with a 295 gtx and seen some really amazing speeds. The home page for the cuda-multiforcer is [http://www.cryptohaze.com](http://www.cryptohaze.com). I would like to give a special thanks to Bit Weasil for a great tool and his help with my understanding of CUDA.
Crank:

Crank is the newest tool in Backtracks CUDA arsenal. It is a GPU powered rar archive cracker. It works in the same way as many password tools by specifying a charset and then attacking the file with a brute force. The home page for crank is http://www.crark.net/

Advantages of cRARk are:

1. One of the fastest RAR password recovery software, uses extremely optimized MMX & SSE cod
2. Support of RAR password recovery on NVIDIA GPU using CUDA technology
3. Multi-volume, self-extracting, encrypted header archives support
4. Unique PCL language which is extremely efficient if user remembers any information about a forgotten password

To install the tool: apt-get install crank

The charsets are defined in password.def. Have a look at the syntax of the file to figure out the charset you need to use. For the sake of this document I created a rar archive with a password of “bt4”. Here is a example of an attack.
The entire manual for this tool is available for viewing online at [http://www.crark.net/cRARk.html](http://www.crark.net/cRARk.html). There is a very complete breakdown of how to use the charset file and how to set the other options to achieve maximum results.
Pyrit

What is pyrit?

Pyrit takes a step ahead in attacking WPA-PSK and WPA2-PSK, the protocol that today de-facto protects public WIFI-airspace. The project’s goal is to estimate the real-world security provided by these protocols. Pyrit does not provide binary files or wordlists and does not encourage anyone to participate or engage in any harmful activity. This is a research project, not a cracking tool.

Pyrit’s implementation allows to create massive databases, pre-computing part of the WPA/WPA2-PSK authentication phase in a space-time-tradeoff. The performance gain for real-world-attacks is in the range of three orders of magnitude which urges for re-consideration of the protocol’s security.

Exploiting the computational power of GPUs, Pyrit is currently by far the most powerful attack against one of the world’s most used security-protocols.
Up and running with pyrit:

Pyrit is already included in the backtrack .iso however the cuda core is not. In order to make sure we have the most recent version of both we will need to apt-get them.
Making sure Pyrit is working:

There are a few small tests to run and see if Pyrit is working properly.

**List Cores:**

```
root@bt:~$ pyrit list_cores
Pyrit 0.2.5-dev (C) 2008, 2009 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3

Connecting to storage... connected

The following cores seem available...
#1: 'CUDA-Device #1 'GeForce 8888 GT''
#2: 'CUDA-Device #2 'GeForce 8888 GT''
#3: 'CPU-Core (SSE2)'
#4: 'CPU-Core (SSE2)'
```

This will show us a output of how many video cards and cores are available for pyrit to use. Something interesting to note is that each gpu will use one core of the cpu in order to run at optimal speed. Only the extra cores will be listed. In this example the processor is a intel Q6600 which is a quad core cpu but
since there are two video cards each one of the uses a cpu core therefore only the other two are listed as being available.

**Benchmark:**

The next test is the benchmark. This will show use the occupancy of the cores in a percentage value and it will give us a rough estimate of the amount of PMK/second than pyrite will be using. As you can see in the picture the occupancy of the video cards is 96 and 92 percent. The reason for this loss is because the X-server is running. Running pyrit form the framebuffer will allow you to squeeze every ounce of power out possible. Also just as a side note, running pyrit while the X-server is running will render your desktop pretty much useless until its finished so plan on doing something else while the numbers are crunching.
The complete reference guide for all the pyrite commands can be found at


I will try to highlight a few of the more popular examples here in this document.

**The WPA Wordlist:**

We have created a optimized wpa password list for users to get started with. This was too big for the .iso however we can easily grab it from the repo with apt-get
**Attack_passthrough Mode:**

The first way it can be run is in passthrough mode. The reason this mode is nice is because instead of created bulky tables and writing them to hard disk, Pyrit simply computes the hash’s and pipes them directly into cowpatty. Aircrack-ng does not currently support this option. In order to use this option simply create a command string with the following syntax:

```
root@bt:~# pyrit -e infected -i /pentest/passwords/wordlists/wpawordlist.txt -r infected.cap attack_passthrough
```

```
Password was not found.
```

```
root@bt:~# 
```
Attack-passthrough with Crunch:

Although brute forcing WPA is pretty much useless I will show one way it can be done. If the passphrase was all digits or a phone number this would be a viable option. We can use the tool crunch which is located on the backtrack.iso:

The syntax of the crunch command is:

```
Crunch <min length> <max length> <charset> -o <output file> or | to stdin to pyrite
```

This attack will take a very long time if used with a full charset.
Batch Mode:

Creating tables with pyrit involves a few extra steps but you will have created a table which can be used over and over as long as the essid of the AP is the same.

The basic procedure for this is to import an essid, import a password list and then run the batch command in order to compute the tables.
Now once you have a database made there are many things you can do. You can still export to cowpatty or airolib-ng format just like always. I am going to skip these two options and highlight the newer ones. Pyrit now has the ability to attack the passwords without any other program. Once you have a database created you can attack the capture directly without any export necessary.

```
root@bt: ~ Shell - Konsole
Session  Edit  View  Bookmarks  Settings  Help

root@bt: # pyrit -e infected -r infected.cap attack_batch
Pyrit 0.2.5-dev (C) 2008, 2009 Lukas Lueg http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3

Connecting to storage... connected

Parsing file 'infected.cap' (1/1)... 928 packets (928 802.11-packets), 8 APs

Picked AccessPoint 80:1d:7e:64:9a:7c automatically...
Attacking handshake with Station 80:13:82:a7:dd:53
Tried 993916 PMKs so far (100.0%); 554038 PMKs per second.

The password was not found.
root@bt: #
```
Analyze:

Another nice new feature of Pyrit, is the ability to analyze the file by itself.

```
pyrit -r infected.cap analyze
```

Analyze can parse one or more packet-capture files in cap and pcap-format and try to detect Access-Points, Stations and EAPOL-handshakes. For example:

```
pyrit -r "test*.pcap" analyze
```

The suffix ‘handshake found’ is appended to the Station's BSSID if the communication between the Access-Point and the Station seems to include a valid EAPOL-handshake.
Strip:

The last thing I would like to highlight is the strip feature. I have noticed that the one downfall of Pyrit is that the handshake detection is still a little lacking when compared with aircrack-ng or Cowpatty. The times when I have had problems is when there are multiple handshakes in on file. So there is a new strip feature in Pyrit to strip the file. Look back at the capture from the analyze section. Although there is only one handshake in there are some other access points we could get rid of.
Building aircrack-ng with CUDA support:

This is still under heavy development so it is not yet been added to the backtrack repositories however it deserves mentioning. Aircrack can be built with a switch to add GPU acceleration. In order to do this we need to grab aircrack from svn. You must have the toolkit and the sdk installed to be able to build this.

svn co http://trac.aircrack-ng.org/svn/branch/aircrack-ng-cuda aircrack-ng-cuda

Next we will build it like normal but it needs a few extra arguments

root@bt~# cd aircrack-ng-cuda

root@bt:~/aircrack-ng-cuda~#CUDA=true make

root@bt:~/aircrack-ng-cuda~#make CUDA=true sqlite=true unstable=true install

Test to ensure everything is working, run aircrack on the test wpa-psk capture file, with the included wordlist:

root@bt~# cd src

root@bt~# ./aircrack-ng -p 1 ../test/wpa.cap -w ../test/password.lst
The -p switch is what adds the CUDA function to aircrack-ng. I have tested the tool and it does work but like I said its underdevelopment and could use some optimization. In my testing pyrit was still quite a bit faster however your milage may vary.

Special thanks to Zermelo and fnord0 for testing and posting the results of this tool.

**Outro:**

I have only scratched the surface of the power of CUDA in this document. I sincerely hope that programmers will continue to create penetration testing tools which use the awesome power of CUDA. I truly believe that parallel GPU computing is the future for all the repetitious tasks we as penetration testers have to do daily including passwords attacks, brute force attacks and fuzzing. I hope to develop some applications of my own as I become more and more adept at CUDA programming.
Useful Links:

http://www.nvidia.com/object/cuda_home.html

http://forums.nvidia.com

http://impact.crhc.illinois.edu/ftp/report/impact-08-01-mcuda.pdf

https://visualization.hpc.mil/wiki/GPGPU


http://pyrit.wordpress.com/

http://www.cryptohaze.com/

http://backtrack-linux.org/forums/

http://www.offensive-security.com/blog/
Special Thanks:

There is no way to thank everyone who has helped out with this stuff but I will try to name a few:

The entire Backtrack-dev team, thebaron, ebfe, Synok, Gommet and the rest of the guys in #cuda IRC Channel. Zero_Chaos and Grimmlin from the Pentoo team and the guys from the Net-Sploit Team. I would also like to give a special thanks to Edgan for the long tireless hours we worked together to understand this stuff and anyone else who I have forgotten because I have ADD.

Anyone who wishes to contact me may do so on our IRC channel #backtrack-linux on the freenode network. Please do not contact me via email with silly questions.

Thanks <3 Pureh@te