4-2017

A Study on the Security of Password Hashing Based on GPU Based, Password Cracking using High-Performance Cloud Computing

Parves Kamal
pkamal@stcloudstate.edu

A Study on the Security of Password Hashing Based On GPU Based, Password Cracking Using High-Performance Cloud Computing

by

Parves Kamal

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree, of

Master of Science

in Information Assurance

.

March,2017

Starred Paper Committee:
Dr, Dennis Guster, Chairperson
Dr. Jim Q. Chen
Dr.Mahbub Hossai

Abstract

In This paper the current security of various password hashing schemes that are in use today will be investigated through practical proof of concept – GPU based, password hash dump cracking using the power of cloud computing. The focus of this paper is to show the possible use of cloud computing in cracking hash dumps and the way to countermeasures them by using secure hashing algorithm and using complex passwords.

Table of Contents

Pages

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Introduction

The most common means of authentication scheme are password based authentication system [1]. An employee uses multiple passwordson adaily basis for all the applications and systems that he/she might be working on for the employer. Businesses spend atremendous amount of money for not only storing these passwords but also for securing the storage of these passwords. Especially when organization deals with a huge number of customers;it's very hard for them to create, maintain and distribute these passwords across the network for authentication, authorization or accounting purposes. Thus, Passwords based authentication system possesses many security problems into rather relatively secured existing infrastructures[2].

To overcome the possible security concerns with storing and distributing the password across the network, the password is often run against the cryptographic hash function to get the equivalent digest of the password which is stored along with the user's other credentials in the database. When users try to login with the password, the input is calculated by the same hash function to compare with the digest of the same password that has been stored in the databases.

One of the properties of the cryptographic hash function is its irreversible one-way function which means it's nearly impossible to get back the password from the digest itself. Then again, many of the commonly used hashing functions like MD5, SHA-1, etc. have been developed during the mid-nineties. One of the weaknesses of most widely used hash function MD5 is that the attacker can create two identical digests for two different inputs which in cryptography field is called Hash collision [3]. In fact, the possibility of an adversary of finding the password from the hash dump is proportional to the amount of the work he/she puts in and the ability to predict the password characteristics distribution. Moreover, with the advent of the cloud computing and

new powerful Graphics Processing Unit (GPU), the attacker is now well equipped than ever to decipher the passwords from the hash at their will.

Since last decades there has been significant development going on in the field of Graphics Processing Unit (GPU). The GPU is very suitable for performing parallel tasks as well as calculating floating point related problems. Modern GPU based, password cracking is tentimes faster than Central Processing Unit (CPU) based password cracking [4]. In my paper, I will be showing how an attacker can leverage existing cloud services to crack passwords from sample password hash dumps using the high-performanceGPU-based computing resources.

1.2 Problem Statement

Many organizations are still using vulnerable hashing functions like MD5/SHA-1 for storing their user's passwords. These hashing functions are inherently weaker and susceptible to many hashing related attacks. Also, the complexity of passwords is still lenient across many organizations,hence making it possible for the attacker to perform dictionary based rainbow attacks against those hashed passwords. Adding salting and choosing a relatively slow hashing function ensuresthat such exhaustive password cracking techniques in the near future will be still impractical [5].

1.3 Nature and Significance of the Problem

Due to the recent hacking and public disclosure of private information (User's passwords) from several big profile organizations like LinkedIn, E-harmony and Yahoo within last 5 years raises the serious questions of not only the security of the authentication systems in these high

profile organizations but also the security aspects of their password storing techniques in their databases.

Hence, I am taking this opportunity in my paper to show how effective theGPU based, password cracking technique is against the hashing techniques and provides insighton why choosing strong, complex passwords along with slow computers hashing function will keep the attackers at bay while leveraging GPU processing power of the cloud computing resources.

1.4 Objective of the Study

The objectives of this paper are to show the effectiveness of a GPU based,password cracking by using cloud computing against latest password hashing techniques. By doing so, this paper providesa recommendation on how to store passwords more securely, which makes any possible distributed GPU based, password cracking techniques inefficient.We achieve these goals in following ways:

- Present the literature review of the current password hashing techniques and identifying their advantages and disadvantages.
- Provide insight into different password cracking techniques and why GPU based,password cracking using high-performancecomputing in thecloud is viable.
- Demonstrate the effectiveness of a GPU based, password cracking of hashed dump on cloud computing.
- Providing comparisons amongst the more cryptographically strong hashing techniques.
- Come to a conclusion and suggestions on the importance of using salt in hashing as well as the strong password selection.

1.5 Study Questions/Hypotheses

By the end of this paper, we should be able to answer following questions:

How does the GPU affect the performance of exhaustive password cracking techniques in cloud computing?

To answer the above question, the following sub-questions will help to answer the research inquiry:

- What are the Password hashing attacks?
- What are the advantages of GPU based, password cracking over CPU based password cracking?
- How to implement GPU based, cracking in cloud computing?
- Why use computationally slow hash function and salt in hashing?
- The Importance of using non-human readable passwordsrather than human-memorable passwords.

1.6 Definition of Terms

Acronym	Description
GPU	Graphics Processing Unit
CPU	Central Processing Unit
MD5	Message Digest Algorithm 5 (1992)
Hash	Cryptographic fixed-size value from arbitrary input
Collision	Two different input yields to the same output hash value
API	Application programming interface
GPGPU	General-Purpose Graphics Processing Unit
GFLOPS	(Giga Floating Point Operations Per Second)
OpenCL	Open Computing Language
CUDA	Compute Unified Device Architecture

1.7 Summary

This chapter summarizes the research problem statements, objectives,and research questions. In the next chapter, we will explore the background related to the problem as well as the literature review.

Chapter 2: Background and Review of Literature

2.1 Introduction

This chapter will provide the required theoretical background necessary to understand the research topics to understand better the research question that we are going to solve. Also, this chapter will provide the related work that is regarding the GPU based, password cracking and on password hashing security.

2.2 Background Related to the Problem

This section will provide necessary definitions and concepts related to password hashing and their attacks, characteristics of strong passwords and their expected cracking time, the properties of thesecure hash function, performance differences between GPU and CPU-based password cracking and cloud computing.

2.2.1 What is aHash?

The process of taking an arbitrary length inputand converting it to a fixed-length output value by integratingthrough a cryptographic hash function is called hashing, and the output value is called the hash value as shown in the figure below. The property of cryptographic hash functions is they are aone-wayfunction, and there is no way of deducing the input value by reverting from the output hash value [6].

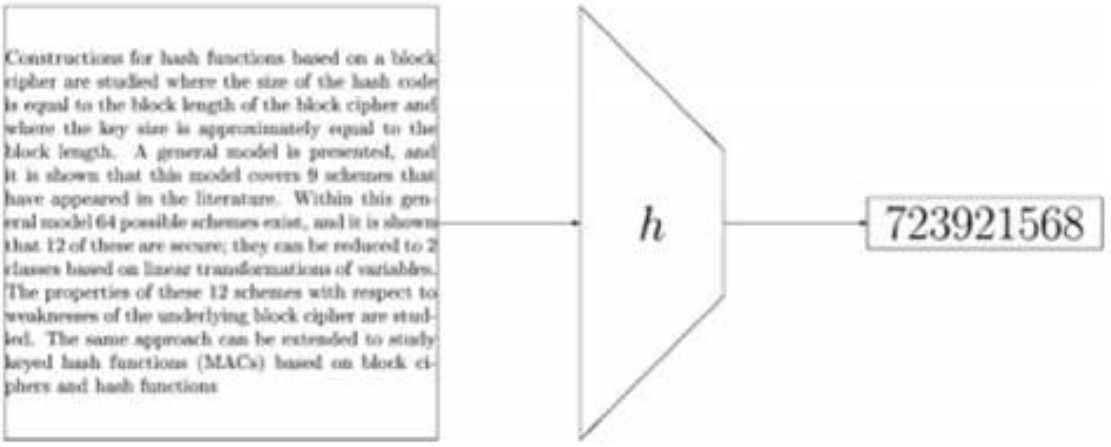


Figure 1.Hashing (h) Function in operation [7].

Because of its irreversible features, hashing is very useful for storing thepassword in the authentication server. This allows not only protecting customer's privacy but also allows theserver to authenticate user's login information without storing the user's password. Let's say for example the cryptographic hash value of the word 'parves' using MD5 algorithm is:

MD5(parves) = cf7cccd2e366698244ac5891da31bb82

2.2.2 Different Hashing Algorithm:

In the following table below the function of a different hashing algorithm that is in use at largeare shown:

Table 1

Comparison between different hash algorithms[8].

Algorithm	Word Size	Block Size	OutputSize	Rounds	Collision Found
MD-2	32	128	128	18	YES
MD-4	32	512	128	48	YES
MD-5	32	512	128	64	YES
SHA-0	32	512	160	80	YES
SHA-1	40	512	160	80	YES
SHA-2	56/64	512/1024	224/256/384/512	64/80	THEORETICAL
SHA-3	64	1152/1088 832/576	224/256/384/512	24	NO

2.2.3 Characteristics of Secure Hash Function

Any hash function needs to have the following two properties:

- **Fixed Output:** Hash function always needs to theoutput of a fixed amount of output irrespective of the input length size. If a hash function h outputs to 64 bits of value, then an input message of m=22 bits and n=50 bits should always compute to thesame length of the64-bit hash value .i.e.h(m) → 64 bits' hash value and h(n)→ 64 bits' hash value.

- **Easy Computation function**: Hash function should be implemented as such that for any input y, h(y) should be easy to compute [9].

To be qualified as a cryptographically secured hash function, it must meet the following three characteristics in their function:

- **Preimage Resistance**: For all the hashes of x, it is extremely difficult to find the message m which is hashed to x.
- **Second-preimage Resistance**: For any given message x, it's almost very highly unlikely to find another message y which has the same hash value like message x, i.e. $h(x) \neq h(y)$ where h is any hash function and $x \neq y$
- **Collision Resistance**: It should be very difficult to find two different message computes to the same hash value. In this type of attack, the attacker is free to choose any two different messages and find the similar hash value. i.e. $h(x) = h(y)$

2.2.4 Password Hash Cracking Techniques

Password hash cracking is amethod of attacking dumped hash to find flaws in the underlying secured hash characteristics that we discussed in theprevious section to find the message that computer to the same hash value. The attacker, once they get hold of the hash file by getting the unauthorized access to the network, usually copies the hashed password file and performs one of the following attacks.

- **Exhaustive Attack:**

  This attack involves trying every possible combination of characters within character sets. Since it looks for every possible combination, the success rate is 100% of finding the correct combination given the time, and the cost is out of consideration. Also,

themostcryptographic system in use today uses very large space of time, making the exhaustive search impractical to perform against. The exhaustive attack is used in two of the following ways [10]:

- ✓ **Known-Plaintext only attacks**: In this attack, only the perpetrator knows plain text ciphertext both. Then he tries to discover the key that encrypts the plaintext for example.
- ✓ **Ciphertext-only attacks**: The attacker only knows the ciphertext, and he tries to findthe corresponding key or plaintext by going through every combination of the keys.

Password hashing crack technique is only possible in cipher text theonly method as every hash function is aone-way function.However, with a single digit increase in password length makes the exhaustive search iteration increase exponentially, making the exhaustive search attack impractical for the attacker, especially when long and complex alphabetical characters are chosen [11].

- **Dictionary Attack**

A Dictionary attack is avery effective attack if theuseruses some human-memorable password for their login credentials and the attacker tries a list of common words and expressions used in any language for example in English to find the password. Users tend to use common or simple passwords [12] across many platforms, so that can be easily recalled and if it's the case then performing dictionary attacks is highly successful. But some simple modification to those common words can make a dictionary attack highly unsuccessful too [13].

- **Rainbow table Attack**

  Rainbow table Attacks involve looking up the precomputed hash tables and the corresponding key value to find any matching hash. This type of attack, though, has limited combination to look up for but when working within its constraints, it takes less time decrypting hashed password than those two other attacks mentioned earlier [14].

2.2.5 Characteristics of Strong Passwords

We will discuss in the following few sections the characteristics of strong passwords, and we will define its strength based on the exhaustive brute force attack time it takes to crack those passwords. We will then introduce the concept of CPU and GPU based, password cracking with cloud computing to end the chapter.

There are many characteristics that make a password easy to crack using exhaustive attacks. Some of the characteristics are:

- Passwords are based on common dictionary words
- Passwords are Easily guessable
- Passwords are relatively short in length, making it possible to brute force attack easily
- Passwords have some sorts of thepattern which is easy to deduce. e.g. abc123, XYZ, 46824682 etc.
- Passwords have been repeating characters like abab11 or xy12x. People Use the passwords with repeating characters so that they can remember.

To better understand, let'sshow it more graphical way. The following pictures are some of the chosen passwords with acombination of characters, numbers, and symbols to check the strength of the passwords based on the characteristics of the characteristics above of strong passwords.

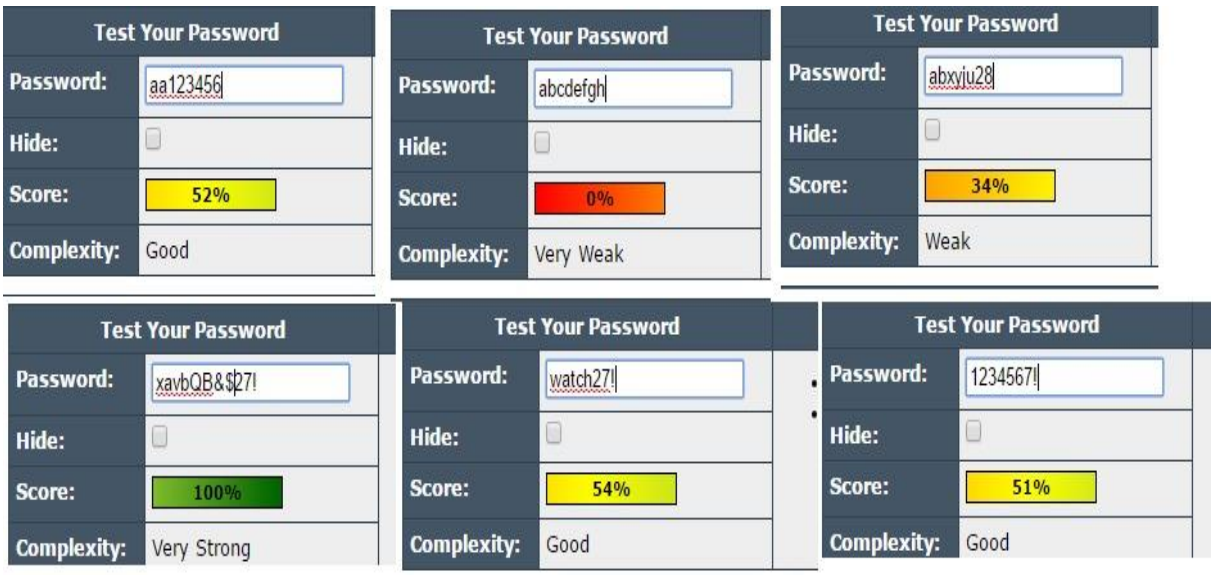


Figure 2. Example of Some of the weak and strong passwords [15].

So as we have seen even if all the passwords above were eight characters long, using memorable words like awatch or if using characters that come in sequences significantly make the password strength very low while using random sequences of characters makes passwords relatively stronger. It's very hard to get real random numbers, and for humans, we are not well equipped to remember random numbers. In fact, ahuman can only remember random numbers up to 7±2 characters [16]. Having passwords with arandom sequence of numbers with at leastone capital letter, number, aSpecial character from large key space makes password very strong as shown below in the above figure.

One of the methods used to check the strength of the password is based on the entropy. Entropy depends on the length of the passwords and the key space used. Entropy can be calculated using the formula, $\log 2\ (n)$ [17],where n represents the number of characters in key space.

2.2.6 CPU-Based Versus GPU-Based Password Cracking Performance:

The CPU has traditionally been used for general purpose computing. Usually,the CPU has limits on how many processing cores it can accommodate. To overcome these CPU's is hyper threading technology to compensate whereas the GPU has many more cores compared to its similarly priced CPU counterpart.

GPU also works as Single instruction, but Multiple Data computations or known as (SIMD) whereas CPU's work as Single Instruction, Single Data computations or known as (SISD). This makes GPU largely suited for password cracking. The difference between CPU'S and GPU'Scan regard performance based on single precision floating point number is clearly shown in the figure below:

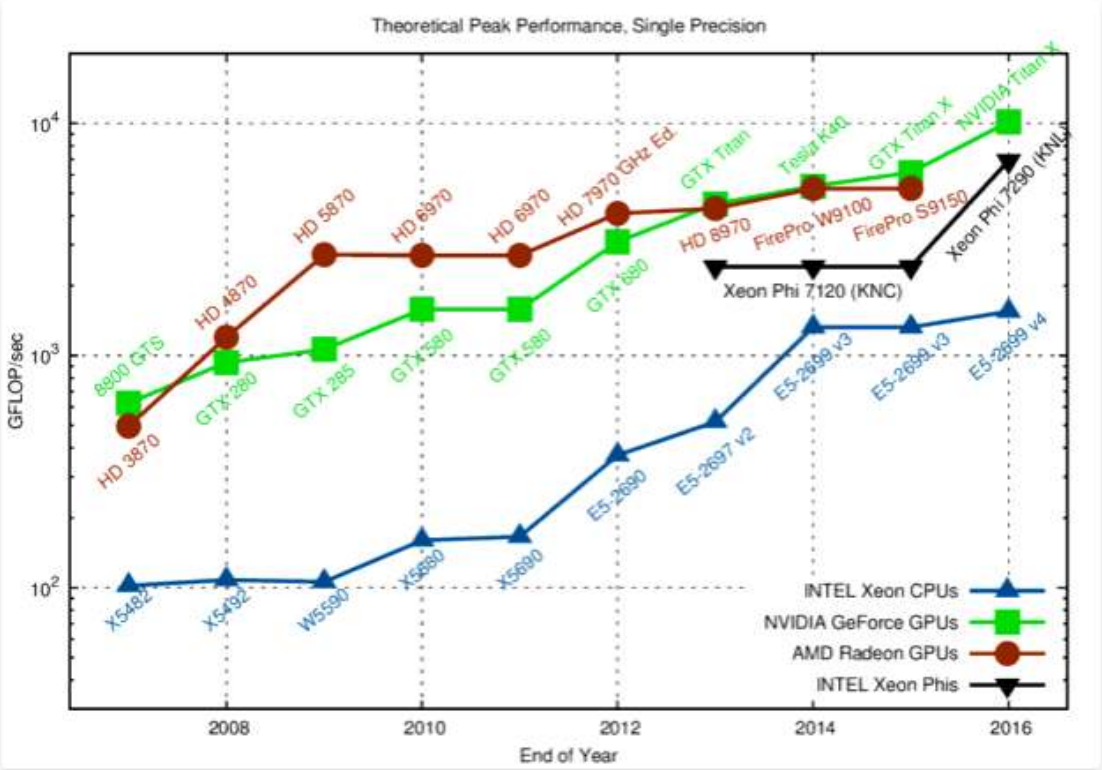


Figure 3. Theoretical peak GFLOP/Sec between CPU'SvsGPU'S [18]

Part of the reason behind the GPU'S power over CPU is for the recent increase in GPU'S power in comparison with the CPU'S one. AS we know from Moore's law CPU'S power doubles once in 18 months, whereas GPU'S power is doubling four times at the same time [19].

Here are the few reasons highlighted behind the GPU'S strength over the CPU:

- CPU's havefewer cores compared to GPU's cores as shown below

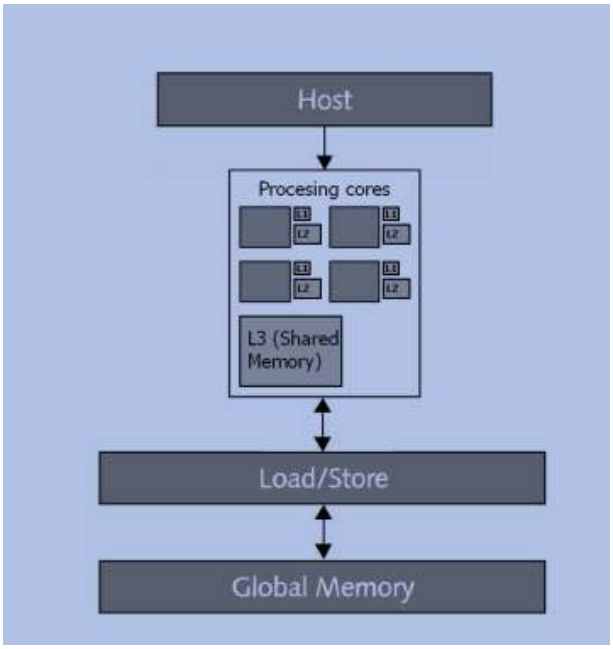


Figure 4. CPU Architecture.

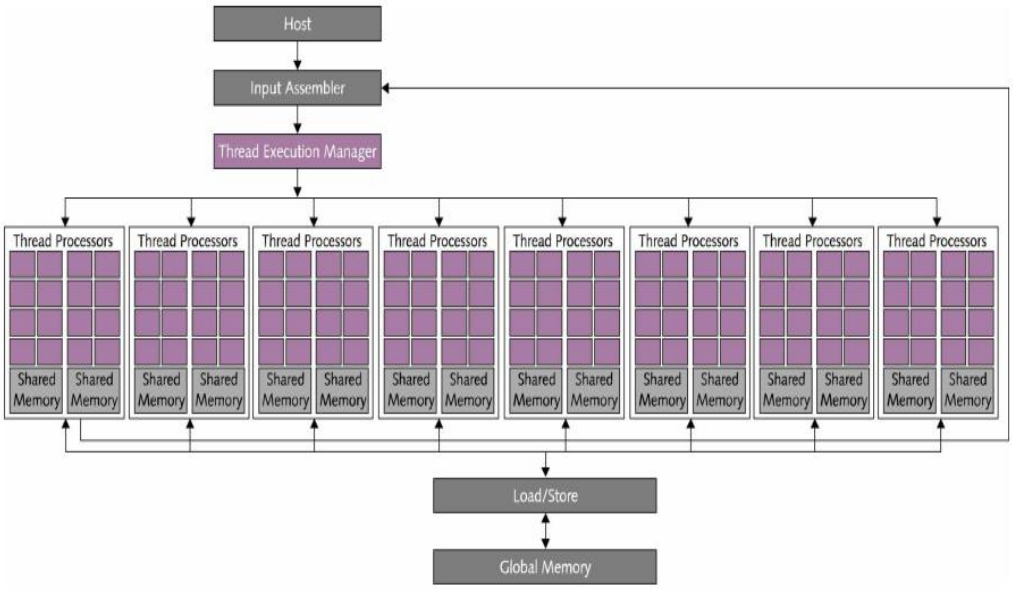


Figure 5. GPU Architecture.

- GPU'Suseless cache than CPU because of the way GPU'S make use of the pipelines and the static operations [20].
- CPU's need to preserve the backward compatibility features to work with old instruction sets, whereas the GPU's are leaving their old instruction sets making it possible to exploit SIMD features whereas even though the new features SIMDv5 is used by the new CPU's, it still needs to be backward compatible with old instruction sets. There are a lot of different GPU architectures, but all these differences are taken care of at the hardware level, leaving the software developer out of hassle.
- The reason GPU's havealarge amount of GFLOPS (Giga Floating Point Operations Per Second) because programming application can exploit the power of GPU's is easier because of the introduction of Unified Device Architecture (CUDA) and Open Computing Language (OpenCL). CUDAis used for parallel computing, which developer exploits while writing programs for GPU's. OpenCL is an open source language framework, making it possible to write code applicable for any GPU's architecture.
- GPU's needs to make use of the cache and the register shown in fig-5 that is on board to take advantages of its processing power [21].

2.2.7 Cloud Computing

Cloud computing is rapidly provisioned on demand configurable resources that can be shared with minimal management effort or service overhead [22]. There are two types of cloud – public and private. Public cloud is a cloud service offered by thecloud. APrivate cloud is exclusively provisioned for acertain group of users. Because of the flexibility and the cost effectiveness every

company is moving their infrastructure to the cloud now. There are many public cloud companies like AWS, Rackspace, Google Cloud, Microsoft Azure provide.

Properties	Amazon EC2	Google AppEngine	Microsoft Azure	Manjrasoft Aneka
Service Type	IaaS	IaaS – PaaS	IaaS – PaaS	PaaS
Support for (value offer)	Compute/Storage	Compute(web applications)	Compute/Storage	Compute
Value Added Provider	Yes	Yes	Yes	Yes
User access Interface	Web APIs and Command Line Tools	Web APIs and Command Line Tools	Azure Web Portal	Web APIs, Custom GUI
Virtualization	OS on Xen Hypervisor	Application Container	Service Container	Service Container
Platform (OS & runtime)	Linux, Windows	Linux	.NET on Windows	.NET/Mono on Windows, Linux, MacOS X
Deployment Model	Customizable VM	Web apps (Python, Java, JRuby)	Azure Services	Applications (C#, C++, VB, ...)
If PaaS, ability to deploy on 3rd party IaaS	N.A.	No	No	Yes

Figure 6. Cloud Computing Offered solutions and their comparison [23].

Spinning up a virtual machine in the Amazon AWS with Intel Xeon-based G series computing along with NVidia's K520 (1536 cores) cost about $0.02 to $2.87 per hour making it very feasible for someone to run parallel GPU-based, password cracking on the password hash dump. In my paper I will demonstrate how effective is GPU based, password cracking using cloud platform.

2.3 Literature Related to the Problem

Thompson in his paper has shown how GPU's can be used for usual computing task other than graphically intensive work [24] whereas Cook in his paper shown GPU's are good for solving cryptography related work too [25]. In the beginning, it was hard to make any application that could take advantage of GPU's because of the lack of API's and supports. Now with the advent of CUDA, OpenCL platform, APIs for GPU's become widely available. Using GPU's performance on cryptographic computation has been under review by many researchers. Yang

and his colleague have shown how GPU's can outperform high-performance CPUs in symmetric cryptographic computations [26]. In Asymmetric encryption strength has also been reviewed by many researchers suing GPU's [27,28,29]. Bernstein has shown how NVidia GTX 295 can be used to break ECC cryptosystem by calculating up to 481 million modular multiplications per second in his paper [30]. Hash functions like MD5 and Blowfish have also been tested with GPU processing power, outperforming the CPU's significantly [31,32].

2.4 Literature Related to the Methodology

Graphical Processing Unit or GPU is now lots used for general purpose computing or better known as GPGPU than rather using it as to drive graphics.

With the advent of CUDA and OpenCL framework researcher are putting the hash security to thetest by exploiting the power of the GPU to crack them using parallel processing power. R. Zhang and his colleagues showed the method to crack MD5 hash using CUDA and reached the speed of 223 Mbps [33]. Another researcher compared decryption software John, the Ripper against cracking software based on OpenCL and found 17-times faster speed [34]. In other research, the authors implemented MD5 decryption methods using Tianhe-1A using CUDA to reach calculation speeds up to 18 billion keys per seconds [35]. Oclhashcat happens to be the multiplatform world's fastest password cracker which is GPUGPG based open source free hash cracker with speeds of up to 8511 mc/s and 2722 mc/s for MD5 and SHA-1 hash respectively. TheGPU has also been used for better and faster implementation of hashing algorithm as well. In one study researcher, implemented MD5 decryption algorithm using GPU cluster and gain 100 times faster performance in comparison to CPU [36]. Moreover, researchers have

madeacomparison toaparallel version of MD5 on NVidia's GPU of which the results are whom in the following table:

Table 2

MD5 is cracking performance comparison using GPU [37].

Platform	Language	Performance
NVidia Geforce 9600 GT	CUDA	223 Mbps
Nvidia GTX295	OpenCL	76.6 Mbps
Nvidia GTX9800+	CUDA	516 Mbps
AMD HD7970(1G)	OpenCL	507.3 Gbps
AMD HD7970 (925 M)	OpenCL	409.9 Gbps

As shown above,research has been done on cracking MD5 hashing using GPU's and relatively less onan SHA-1 hashing algorithm. Many of the researchers used open source cracker like Oclhashcat on GPU platforms like NVidia or AMD using CUDA or OpenCL. In our paper, we will also exploit GPU power that is on offered in the cloud to crack sample password hash dump using Oclhashcat.

2.5 Summary

In this chapter, we covered all the required definitions and background knowledge related to the research and we also covered the literature review related to our research problem and methodology that we will be using. In the following chapter, we will outline our research methodology in detail and the necessary tools and the techniques as well as our test environment in detail.

Chapter 3: Methodology

3.1 Introduction

In this chapter, we will discuss our methodology used in conducting our research.

3.2 Design of the Study

To answer the research questions proposedin section1.4, the following approaches will be followed:

- Literature review: We try to identify the characteristics of strong hashing algorithm as well as the passwords with the data gathered from the proof of concept—password hash cracking in the cloud
- Proof of concept: To show the performance of the GPU on password hash cracking using cloud we will implement GPU based, hash cracking on cloud and we will compare this with CPU based cracking.
- Comparison: We will compare the analyzed data to compare the GPU based, and CPU based, password cracking performance as well as the effect of using secured password hashing algorithm on the cracking performance. We will also analyze the reason behind the password hash cracking effectiveness based on password strength.

3.3 Data Collection

The data will be collected once the test is performed and the benchmark report, as well as the generated passwords, will be the source of the data which will later be analyzed.

3.4 Tools and Techniques

Multi-GPU based OclHashcat on Amazon AWS EC2 on CUDA based NVIDIA Tesla GPU to crack sample password hash of MD5,SHA-256, and Bcrypt.

3.5 Hardware and Software Environment

The following hardware and software will be used in conducting my research. The setup was done in AWS (Amazon Web Service) cloud. The exact software and Hardware details are in the table below. The test will be conducted in cracking the password hash on both GPU and CPU instances (Machine).

Table 3

Hardware and Software Requirements.

GPU Test Machine	
Software	**Hardware**
Ubuntu 16.04 with NVIDIA GRID and TESLA GPU Driver	Intel Xeon E5-2670 (Sandy Bridge) Processors
oclHashcat is a GPGPU-based multi-hash cracker	P2 instances provide up to 16 NVIDIA K80 GPUs, 64 vCPUs and 732 GiB of host memory, with a combined 192 GB of GPU memory, 40 thousand parallel processing cores, 70 teraflops of single precision floating point performance, and

	over 23 teraflops of double precision floating point performance. P2 instances also offer GPU Direct™ (peer-to-peer GPU communication) capabilities for up to 16 GPUs, so that multiple GPUs can work together within a single host
	vCPU - 32 Ram – 488 GPU -8 GB SSD- 300GB

CPU Test Machine	
Software	**Hardware**
Ubuntu 16.04 - - with Updates HVM-1602	Intel Xeon E5-2666 v3 2.9 GHz
oclHashcat is a CPU-based multi-hash cracker	High-frequency Intel Xeon E5-2666 v3 (Haswell) processors optimized specifically for EC2 EBS-optimized by default and at no additional cost Ability to control processor C-state and P-state configuration on the c4.8xlarge instance type
	VCPU-8 RAM – 15GB SSD-30GB

3.6 Installing oclHashcaht and Benchmarking of the Performance of oclHashcat on Both CPU and GPU

First, the installation of the ocHashcat multi-GPU based, hash cracker with the following steps after ssh into our cloud Linux machine on both CPU and GPU test machine.

```
sudo apt -y update && sudo apt -y upgrade
sudo apt install -y p7zip-full build-essential linux-image-extra-virtual
linux-source

echo options nouveau modeset=0 | sudo tee -a /etc/modprobe.d/nouveau-
kms.conf
sudo update-initramfs -u

# to activate latest kernel
sudo reboot

ssh -i keyfile.pem ubuntu@<ip>
sudo apt install linux-headers-`uname -r`

sudo mkdir -p /data
sudo chown ubuntu /data
cd /data

# latest driver here: http://www.nvidia.com/object/unix.html
wget http://us.download.nvidia.com/XFree86/Linux-x86_64/375.26/NVIDIA-
Linux-x86_64-375.26.run
chmod 755 NV*.run
sudo ./NV*.run
# accept license, install, enter, enter

lsmod | grep nvidia
# if not loaded, do extra reboot

# optimizations from AWS blog
sudo nvidia-smi -pm 1
sudo nvidia-smi -acp 0
sudo nvidia-smi --auto-boost-permission=0
sudo nvidia-smi -ac 2505,875

# Download and install Hashcat
wget http://hashcat.net/files/hashcat-3.40.7z
7za x hashcat-3.40.7z
Cd hashcat-3.40

./hashcat64.bin -b
```

Then the benchmark test was run on the different hashing algorithm and the full benchmark results are as follows:

GPU Test Machine Benchmark Results:

root@ip-172-31-43-198 hashcat-3.30]# ./hashcat64.bin –b

OpenCL Platform #1: NVIDIA Corporation

==========

Device #1: Tesla K80, 2047/11439 MB allocatable, 13MCU

Device #2: Tesla K80, 2047/11439 MB allocatable, 13MCU

Device #3: Tesla K80, 2047/11439 MB allocatable, 13MCU

Device #4: Tesla K80, 2047/11439 MB allocatable, 13MCU

Device #5: Tesla K80, 2047/11439 MB allocatable, 13MCU

Device #6: Tesla K80, 2047/11439 MB allocatable, 13MCU

Device #7: Tesla K80, 2047/11439 MB allocatable, 13MCU

Device #8: Tesla K80, 2047/11439 MB allocatable, 13MCU

Hashtype: MD4

Speed.Dev.#1.....: 8466.9 MH/s (51.48ms)

Speed.Dev.#2.....: 8501.4 MH/s (51.27ms)

Speed.Dev.#3.....: 8445.6 MH/s (51.61ms)

Speed.Dev.#4.....: 9079.5 MH/s (48.00ms)

Speed.Dev.#5.....: 8212.8 MH/s (53.07ms)

Speed.Dev.#6.....: 8906.6 MH/s (48.94ms)

Speed.Dev.#7.....: 8260.1 MH/s (52.77ms)

Speed.Dev.#8.....: 9072.7 MH/s (48.04ms)

Speed.Dev.#*.....: 68945.7 MH/s

Hashtype: MD5

Speed.Dev.#1.....: 4551.7 MH/s (95.80ms)

Speed.Dev.#2.....: 4440.1 MH/s (98.20ms)

Speed.Dev.#3.....: 4427.0 MH/s (49.20ms)

Speed.Dev.#4.....: 4626.7 MH/s (94.24ms)

Speed.Dev.#5.....: 4264.6 MH/s (51.10ms)

Speed.Dev.#6.....: 4521.0 MH/s (96.45ms)

Speed.Dev.#7.....: 4229.8 MH/s (51.53ms)

Speed.Dev.#8.....: 4613.5 MH/s (94.51ms)

Speed.Dev.#*.....: 35674.4 MH/s

Hashtype: Half MD5

Speed.Dev.#1.....: 3223.9 MH/s (67.60ms)

Speed.Dev.#2.....: 3156.0 MH/s (69.07ms)

Speed.Dev.#3.....: 3109.5 MH/s (70.10ms)

Speed.Dev.#4.....: 3309.7 MH/s (65.83ms)

Speed.Dev.#5.....: 2947.0 MH/s (73.94ms)

Speed.Dev.#6.....: 3307.7 MH/s (65.88ms)

Speed.Dev.#7.....: 3075.6 MH/s (70.88ms)

Speed.Dev.#8.....: 3311.1 MH/s (65.82ms)

Speed.Dev.#*.....: 25440.4 MH/s

Hashtype: SHA1

Speed.Dev.#1.....: 1993.0 MH/s (54.46ms)

Speed.Dev.#2.....: 1940.6 MH/s (55.91ms)

Speed.Dev.#3.....: 1972.2 MH/s (55.03ms)

Speed.Dev.#4.....: 2045.6 MH/s (53.03ms)

Speed.Dev.#5.....: 1937.4 MH/s (56.02ms)

Speed.Dev.#6.....: 1999.5 MH/s (54.26ms)

Speed.Dev.#7.....: 1872.4 MH/s (57.97ms)

Speed.Dev.#8.....: 2024.6 MH/s (53.61ms)

Speed.Dev.#*.....: 15785.3 MH/s

Hashtype: SHA256

Speed.Dev.#1.....:   815.1 MH/s (66.79ms)

Speed.Dev.#2.....:   815.5 MH/s (66.76ms)

Speed.Dev.#3.....:   811.9 MH/s (67.05ms)

Speed.Dev.#4.....:   874.6 MH/s (93.39ms)

Speed.Dev.#5.....:   810.0 MH/s (67.27ms)

Speed.Dev.#6.....:   836.2 MH/s (65.17ms)

Speed.Dev.#7.....:   797.3 MH/s (68.34ms)

Speed.Dev.#8.....:   847.4 MH/s (64.30ms)

Speed.Dev.#*.....: 6608.0 MH/s

Hashtype: SHA384

Speed.Dev.#1.....: 174.6 MH/s (78.05ms)

Speed.Dev.#2.....: 174.8 MH/s (77.93ms)

Speed.Dev.#3.....: 175.5 MH/s (77.60ms)

Speed.Dev.#4.....: 175.8 MH/s (77.48ms)

Speed.Dev.#5.....: 175.3 MH/s (76.83ms)

Speed.Dev.#6.....: 175.1 MH/s (77.82ms)

Speed.Dev.#7.....: 174.5 MH/s (78.07ms)

Speed.Dev.#8.....: 175.0 MH/s (77.83ms)

Speed.Dev.#*.....: 1400.6 MH/s

Hashtype: SHA512

Speed.Dev.#1.....: 177.5 MH/s (76.75ms)

Speed.Dev.#2.....: 177.8 MH/s (76.64ms)

Speed.Dev.#3.....: 177.9 MH/s (76.60ms)

Speed.Dev.#4.....: 178.9 MH/s (76.15ms)

Speed.Dev.#5.....: 177.5 MH/s (76.77ms)

Speed.Dev.#6.....: 178.1 MH/s (76.49ms)

Speed.Dev.#7.....: 177.7 MH/s (76.68ms)

Speed.Dev.#8.....: 178.1 MH/s (76.51ms)

Speed.Dev.#*.....: 1423.4 MH/s

Hashtype: SHA-3(Keccak)

Speed.Dev.#1.....: 184.6 MH/s (73.79ms)

Speed.Dev.#2.....: 175.5 MH/s (77.65ms)

Speed.Dev.#3.....: 184.5 MH/s (73.84ms)

Speed.Dev.#4.....: 186.9 MH/s (72.90ms)

Speed.Dev.#5.....: 180.1 MH/s (75.66ms)

Speed.Dev.#6.....: 190.8 MH/s (71.40ms)

Speed.Dev.#7.....: 182.7 MH/s (74.57ms)

Speed.Dev.#8.....: 183.4 MH/s (74.27ms)

Speed.Dev.#*.....: 1468.5 MH/s

Hashtype: SipHash

Speed.Dev.#1.....: 8380.1 MH/s (52.01ms)

Speed.Dev.#2.....: 8436.0 MH/s (51.64ms)

Speed.Dev.#3.....: 8236.4 MH/s (52.91ms)

Speed.Dev.#4.....: 8848.4 MH/s (49.26ms)

Speed.Dev.#5.....: 8010.7 MH/s (54.41ms)

Speed.Dev.#6.....: 8694.2 MH/s (50.13ms)

Speed.Dev.#7.....: 8078.2 MH/s (53.95ms)

Speed.Dev.#8.....: 8825.6 MH/s (49.39ms)

Speed.Dev.#*.....: 67509.6 MH/s

Hash type: RipeMD160

Speed.Dev.#1.....: 1218.0 MH/s (89.15ms)

Speed.Dev.#2.....: 1196.2 MH/s (90.77ms)

Speed.Dev.#3.....: 1200.0 MH/s (90.49ms)

Speed.Dev.#4.....: 1236.2 MH/s (87.81ms)

Speed.Dev.#5.....: 1177.3 MH/s (92.22ms)

Speed.Dev.#6.....: 1233.4 MH/s (88.38ms)

Speed.Dev.#7.....: 1120.3 MH/s (48.63ms)

Speed.Dev.#8.....: 1253.8 MH/s (86.94ms)

Speed.Dev.#*.....: 9635.2 MH/s

Hashtype: Whirlpool

Speed.Dev.#1.....: 78468.3 kH/s (85.80ms)

Speed.Dev.#2.....: 78975.2 kH/s (85.24ms)

Speed.Dev.#3.....: 78928.0 kH/s (85.30ms)

Speed.Dev.#4.....: 78865.1 kH/s (85.34ms)

Speed.Dev.#5.....: 78963.1 kH/s (85.25ms)

Speed.Dev.#6.....: 78982.6 kH/s (85.23ms)

Speed.Dev.#7.....: 78080.8 kH/s (86.23ms)

Speed.Dev.#8.....: 79145.9 kH/s (85.07ms)

Speed.Dev.#*.....: 630.4 MH/s

Hashtype: GOST R 34.11-94

Speed.Dev.#1.....: 66313.6 kH/s (101.54ms)

Speed.Dev.#2.....: 66579.7 kH/s (101.12ms)

Speed.Dev.#3.....: 66764.5 kH/s (100.85ms)

Speed.Dev.#4.....: 66243.8 kH/s (101.65ms)

Speed.Dev.#5.....: 66301.9 kH/s (101.55ms)

Speed.Dev.#6.....: 67160.6 kH/s (100.25ms)

Speed.Dev.#7.....: 65520.9 kH/s (102.76ms)

Speed.Dev.#8.....: 67115.1 kH/s (100.32ms)

Speed.Dev.#*.....: 532.0 MH/s

Hash type: GOST R 34.11-2012 (Streebog) 256-bit

Speed.Dev.#1.....: 19925.1 kH/s (170.97ms)

Speed.Dev.#2.....: 19592.7 kH/s (173.87ms)

Speed.Dev.#3.....: 20090.6 kH/s (169.56ms)

Speed.Dev.#4.....: 20300.8 kH/s (167.79ms)

Speed.Dev.#5.....: 19341.0 kH/s (176.13ms)

Speed.Dev.#6.....: 19972.5 kH/s (170.55ms)

Speed.Dev.#7.....: 19413.8 kH/s (175.47ms)

Speed.Dev.#8.....: 20344.7 kH/s (167.44ms)

Speed.Dev.#*.....: 159.0 MH/s

Hash type: GOST R 34.11-2012 (Streebog) 512-bit

Speed.Dev.#1.....: 19861.1 kH/s (171.55ms)

Speed.Dev.#2.....: 19686.2 kH/s (173.07ms)

Speed.Dev.#3.....: 20028.9 kH/s (170.09ms)

Speed.Dev.#4.....: 20083.5 kH/s (169.62ms)

Speed.Dev.#5.....: 19398.3 kH/s (175.65ms)

Speed.Dev.#6.....: 20046.2 kH/s (169.97ms)

Speed.Dev.#7.....: 19641.2 kH/s (173.46ms)

Speed.Dev.#8.....: 20436.0 kH/s (166.72ms)

Speed.Dev.#*.....: 159.2 MH/s

Hash type: DES (PT = $salt, key = $pass)

Speed.Dev.#1.....: 3699.9 MH/s (58.77ms)

Speed.Dev.#2.....: 3647.5 MH/s (59.63ms)

Speed.Dev.#3.....: 3679.8 MH/s (59.09ms)

Speed.Dev.#4.....: 3694.4 MH/s (58.86ms)

Speed.Dev.#5.....: 3607.1 MH/s (60.31ms)

Speed.Dev.#6.....: 3822.5 MH/s (56.91ms)

Speed.Dev.#7.....: 3789.0 MH/s (57.43ms)

Speed.Dev.#8.....: 3779.2 MH/s (57.57ms)

Speed.Dev.#*.....: 29719.3 MH/s

Hash type: 3DES (PT = $salt, key = $pass)

Speed.Dev.#1.....: 250.1 MH/s (54.41ms)

Speed.Dev.#2.....: 256.2 MH/s (53.14ms)

Speed.Dev.#3.....: 259.0 MH/s (52.54ms)

Speed.Dev.#4.....: 256.1 MH/s (53.14ms)

Speed.Dev.#5.....: 258.5 MH/s (52.66ms)

Speed.Dev.#6.....: 250.6 MH/s (54.31ms)

Speed.Dev.#7.....: 246.5 MH/s (55.23ms)

Speed.Dev.#8.....: 256.0 MH/s (53.17ms)

Speed.Dev.#*.....: 2033.1 MH/s

Hash type: phpass, MD5(Wordpress), MD5(phpBB3), MD5(Joomla)

Speed.Dev.#1.....: 1379.7 kH/s (76.42ms)

Speed.Dev.#2.....: 1366.3 kH/s (77.23ms)

Speed.Dev.#3.....: 1315.0 kH/s (80.29ms)

Speed.Dev.#4.....: 1419.0 kH/s (74.39ms)

Speed.Dev.#5.....: 1284.7 kH/s (82.18ms)

Speed.Dev.#6.....: 1413.9 kH/s (74.65ms)

Speed.Dev.#7.....: 1362.7 kH/s (77.46ms)

Speed.Dev.#8.....: 1417.3 kH/s (74.47ms)

Speed.Dev.#*.....: 10958.5 kH/s

Hashtype: script

Speed.Dev.#1.....: 202.1 kH/s (32.43ms)

Speed.Dev.#2.....: 202.3 kH/s (32.40ms)

Speed.Dev.#3.....: 199.7 kH/s (32.84ms)

Speed.Dev.#4.....: 197.2 kH/s (33.26ms)

Speed.Dev.#5.....: 182.5 kH/s (35.94ms)

Speed.Dev.#6.....: 150.7 kH/s (43.59ms)

Speed.Dev.#7.....: 141.1 kH/s (46.55ms)

Speed.Dev.#8.....: 151.1 kH/s (43.46ms)

Speed.Dev.#*.....: 1426.6 kH/s

Hashtype: PBKDF2-HMAC-MD5

Speed.Dev.#1.....: 1427.7 kH/s (60.61ms)

Speed.Dev.#2.....: 1413.8 kH/s (61.20ms)

Speed.Dev.#3.....: 1430.8 kH/s (60.40ms)

Speed.Dev.#4.....: 1484.7 kH/s (58.24ms)

Speed.Dev.#5.....: 1374.1 kH/s (62.99ms)

Speed.Dev.#6.....: 1486.3 kH/s (58.11ms)

Speed.Dev.#7.....: 1405.9 kH/s (61.54ms)

Speed.Dev.#8.....: 1484.0 kH/s (58.21ms)

Speed.Dev.#*.....: 11507.3 kH/s

Hashtype: PBKDF2-HMAC-SHA1

Speed.Dev.#1.....: 777.6 kH/s (58.92ms)

Speed.Dev.#2.....: 752.4 kH/s (60.93ms)

Speed.Dev.#3.....: 767.1 kH/s (59.72ms)

Speed.Dev.#4.....: 790.7 kH/s (57.87ms)

Speed.Dev.#5.....: 751.3 kH/s (61.01ms)

Speed.Dev.#6.....: 775.4 kH/s (59.10ms)

Speed.Dev.#7.....: 727.7 kH/s (63.03ms)

Speed.Dev.#8.....: 793.6 kH/s (57.72ms)

Speed.Dev.#*.....: 6135.9 kH/s

Hash type: PBKDF2-HMAC-SHA256

Speed.Dev.#1.....: 292.4 kH/s (84.88ms)

Speed.Dev.#2.....: 286.2 kH/s (86.75ms)

Speed.Dev.#3.....: 290.1 kH/s (85.55ms)

Speed.Dev.#4.....: 298.2 kH/s (83.25ms)

Speed.Dev.#5.....: 283.1 kH/s (87.69ms)

Speed.Dev.#6.....: 295.3 kH/s (84.06ms)

Speed.Dev.#7.....: 289.0 kH/s (85.92ms)

Speed.Dev.#8.....: 298.1 kH/s (83.25ms)

Speed.Dev.#*.....: 2332.3 kH/s

Hashtype: PBKDF2-HMAC-SHA512

Speed.Dev.#1.....: 94553 H/s (67.10ms)

Speed.Dev.#2.....: 93350 H/s (67.97ms)

Speed.Dev.#3.....: 94379 H/s (67.23ms)

Speed.Dev.#4.....: 95693 H/s (66.30ms)

Speed.Dev.#5.....: 89283 H/s (71.92ms)

Speed.Dev.#6.....: 96060 H/s (66.05ms)

Speed.Dev.#7.....: 94247 H/s (67.32ms)

Speed.Dev.#8.....: 95611 H/s (66.36ms)

Speed.Dev.#*.....: 753.2 kH/s

Hashtype: Skype

Speed.Dev.#1.....: 2840.6 MH/s (76.74ms)

Speed.Dev.#2.....: 2912.7 MH/s (74.81ms)

Speed.Dev.#3.....: 2904.7 MH/s (75.03ms)

Speed.Dev.#4.....: 3074.1 MH/s (70.90ms)

Speed.Dev.#5.....: 2846.7 MH/s (76.58ms)

Speed.Dev.#6.....: 2991.6 MH/s (72.84ms)

Speed.Dev.#7.....: 2973.5 MH/s (73.30ms)

Speed.Dev.#8.....: 3109.2 MH/s (70.11ms)

Speed.Dev.#*.....: 23653.1 MH/s

Hash type: WPA/WPA2

Speed.Dev.#1.....: 91985 H/s (70.54ms)

Speed.Dev.#2.....: 90301 H/s (71.86ms)

Speed.Dev.#3.....: 92049 H/s (70.49ms)

Speed.Dev.#4.....: 95516 H/s (67.93ms)

Speed.Dev.#5.....: 88909 H/s (72.98ms)

Speed.Dev.#6.....: 93344 H/s (69.51ms)

Speed.Dev.#7.....: 91694 H/s (70.77ms)

Speed.Dev.#8.....: 94990 H/s (68.31ms)

Speed.Dev.#*.....: 738.8 kH/s

Hashtype: IKE-PSK MD5

Speed.Dev.#1.....: 316.6 MH/s (86.08ms)

Speed.Dev.#2.....: 314.5 MH/s (86.64ms)

Speed.Dev.#3.....: 317.5 MH/s (85.82ms)

Speed.Dev.#4.....: 317.8 MH/s (85.74ms)

Speed.Dev.#5.....: 306.6 MH/s (88.87ms)

Speed.Dev.#6.....: 318.1 MH/s (85.64ms)

Speed.Dev.#7.....: 317.9 MH/s (85.71ms)

Speed.Dev.#8.....: 317.0 MH/s (85.97ms)

Speed.Dev.#*.....: 2525.9 MH/s

Hashtype: IKE-PSK SHA1

Speed.Dev.#1.....: 165.1 MH/s (82.48ms)

Speed.Dev.#2.....: 162.8 MH/s (83.67ms)

Speed.Dev.#3.....: 163.9 MH/s (83.11ms)

Speed.Dev.#4.....: 167.4 MH/s (81.36ms)

Speed.Dev.#5.....: 161.0 MH/s (84.62ms)

Speed.Dev.#6.....: 168.1 MH/s (81.04ms)

Speed.Dev.#7.....: 167.6 MH/s (81.27ms)

Speed.Dev.#8.....: 168.6 MH/s (80.82ms)

Speed.Dev.#*.....: 1324.6 MH/s

Hash type: NetNTLMv1-VANILLA / NetNTLMv1+ESS

Speed.Dev.#1.....: 4720.0 MH/s (92.37ms)

Speed.Dev.#2.....: 4697.4 MH/s (92.82ms)

Speed.Dev.#3.....: 4530.3 MH/s (96.24ms)

Speed.Dev.#4.....: 4710.7 MH/s (92.56ms)

Speed.Dev.#5.....: 4456.6 MH/s (97.81ms)

Speed.Dev.#6.....: 4737.6 MH/s (92.02ms)

Speed.Dev.#7.....: 4407.3 MH/s (98.93ms)

Speed.Dev.#8.....: 4714.3 MH/s (92.49ms)

Speed.Dev.#*.....: 36974.2 MH/s

Hash type: NetNTLMv2

Speed.Dev.#1.....: 295.5 MH/s (92.24ms)

Speed.Dev.#2.....: 291.7 MH/s (93.42ms)

Speed.Dev.#3.....: 287.9 MH/s (94.67ms)

Speed.Dev.#4.....: 294.8 MH/s (92.42ms)

Speed.Dev.#5.....: 293.9 MH/s (92.72ms)

Speed.Dev.#6.....: 294.3 MH/s (92.60ms)

Speed.Dev.#7.....: 294.7 MH/s (92.46ms)

Speed.Dev.#8.....: 294.6 MH/s (92.50ms)

Speed.Dev.#*.....: 2347.3 MH/s

Hash type: IPMI2 RAKP HMAC-SHA1

Speed.Dev.#1.....: 346.0 MH/s (78.76ms)

Speed.Dev.#2.....: 349.4 MH/s (77.99ms)

Speed.Dev.#3.....: 354.6 MH/s (76.83ms)

Speed.Dev.#4.....: 362.8 MH/s (75.09ms)

Speed.Dev.#5.....: 326.6 MH/s (83.42ms)

Speed.Dev.#6.....: 365.4 MH/s (74.57ms)

Speed.Dev.#7.....: 335.4 MH/s (81.25ms)

Speed.Dev.#8.....: 359.4 MH/s (75.82ms)

Speed.Dev.#*.....: 2799.6 MH/s

Hash type: Kerberos 5 AS-REQ Pre-Authtype 23

Speed.Dev.#1.....: 47808.6 kH/s (142.52ms)

Speed.Dev.#2.....: 47191.6 kH/s (144.39ms)

Speed.Dev.#3.....: 47967.8 kH/s (142.05ms)

Speed.Dev.#4.....: 47687.9 kH/s (142.88ms)

Speed.Dev.#5.....: 48071.0 kH/s (141.74ms)

Speed.Dev.#6.....: 47765.8 kH/s (142.63ms)

Speed.Dev.#7.....: 47612.6 kH/s (143.11ms)

Speed.Dev.#8.....: 47767.8 kH/s (142.65ms)

Speed.Dev.#*.....: 381.9 MH/s

Hash type: Kerberos 5 TGS-REP type 23

Speed.Dev.#1.....: 46763.9 kH/s (72.84ms)

Speed.Dev.#2.....: 46724.8 kH/s (72.90ms)

Speed.Dev.#3.....: 46731.8 kH/s (72.88ms)

Speed.Dev.#4.....: 47354.2 kH/s (143.89ms)

Speed.Dev.#5.....: 47632.9 kH/s (143.05ms)

Speed.Dev.#6.....: 47401.7 kH/s (143.75ms)

Speed.Dev.#7.....: 47301.7 kH/s (144.06ms)

Speed.Dev.#8.....: 46954.6 kH/s (72.54ms)

Speed.Dev.#*.....: 376.9 MH/s

Hash type: DNSSEC (NSEC3)

Speed.Dev.#1.....: 714.1 MH/s (76.24ms)

Speed.Dev.#2.....: 742.3 MH/s (73.41ms)

Speed.Dev.#3.....: 773.0 MH/s (70.50ms)

Speed.Dev.#4.....: 781.6 MH/s (69.65ms)

Speed.Dev.#5.....: 736.5 MH/s (73.99ms)

Speed.Dev.#6.....: 764.6 MH/s (71.27ms)

Speed.Dev.#7.....: 747.8 MH/s (72.87ms)

Speed.Dev.#8.....: 776.6 MH/s (70.16ms)

Speed.Dev.#*.....: 6036.6 MH/s

Hash type: PostgreSQL Challenge-Response Authentication (MD5)

Speed.Dev.#1.....: 1430.2 MH/s (75.90ms)

Speed.Dev.#2.....: 1395.0 MH/s (77.80ms)

Speed.Dev.#3.....: 1353.3 MH/s (80.22ms)

Speed.Dev.#4.....: 1487.0 MH/s (73.00ms)

Speed.Dev.#5.....: 1321.7 MH/s (82.15ms)

Speed.Dev.#6.....: 1506.1 MH/s (72.07ms)

Speed.Dev.#7.....: 1323.9 MH/s (82.01ms)

Speed.Dev.#8.....: 1461.1 MH/s (74.30ms)

Speed.Dev.#*.....: 11278.5 MH/s

Hash type: MySQL Challenge-Response Authentication (SHA1)

Speed.Dev.#1.....: 518.0 MH/s (52.52ms)

Speed.Dev.#2.....: 503.8 MH/s (54.01ms)

Speed.Dev.#3.....: 512.9 MH/s (53.06ms)

Speed.Dev.#4.....: 528.4 MH/s (51.50ms)

Speed.Dev.#5.....: 508.2 MH/s (53.55ms)

Speed.Dev.#6.....: 533.2 MH/s (76.66ms)

Speed.Dev.#7.....: 502.3 MH/s (54.18ms)

Speed.Dev.#8.....: 540.8 MH/s (75.58ms)

Speed.Dev.#*.....: 4147.6 MH/s

Hash type: SIP digest authentication (MD5)

Speed.Dev.#1.....: 511.8 MH/s (53.17ms)

Speed.Dev.#2.....: 509.6 MH/s (53.40ms)

Speed.Dev.#3.....: 510.8 MH/s (53.27ms)

Speed.Dev.#4.....: 514.5 MH/s (52.87ms)

Speed.Dev.#5.....: 509.3 MH/s (53.44ms)

Speed.Dev.#6.....: 515.8 MH/s (52.76ms)

Speed.Dev.#7.....: 508.3 MH/s (53.55ms)

Speed.Dev.#8.....: 511.9 MH/s (53.17ms)

Speed.Dev.#*.....: 4091.9 MH/s

Hash type: PostgreSQL

Speed.Dev.#1.....: 4405.2 MH/s (49.47ms)

Speed.Dev.#2.....: 4414.2 MH/s (49.37ms)

Speed.Dev.#3.....: 4267.3 MH/s (51.06ms)

Speed.Dev.#4.....: 4575.1 MH/s (95.31ms)

Speed.Dev.#5.....: 4146.7 MH/s (52.56ms)

Speed.Dev.#6.....: 4615.7 MH/s (94.47ms)

Speed.Dev.#7.....: 4160.1 MH/s (52.39ms)

Speed.Dev.#8.....: 4584.1 MH/s (95.12ms)

Speed.Dev.#*.....: 35168.4 MH/s

Hash type: MSSQL(2000)

Speed.Dev.#1.....: 1734.2 MH/s (62.58ms)

Speed.Dev.#2.....: 1819.9 MH/s (59.64ms)

Speed.Dev.#3.....: 1769.9 MH/s (61.32ms)

Speed.Dev.#4.....: 1871.3 MH/s (58.00ms)

Speed.Dev.#5.....: 1736.4 MH/s (62.52ms)

Speed.Dev.#6.....: 1874.3 MH/s (57.91ms)

Speed.Dev.#7.....: 1754.6 MH/s (61.85ms)

Speed.Dev.#8.....: 1895.5 MH/s (57.25ms)

Speed.Dev.#*.....: 14456.2 MH/s

Hash type: MSSQL(2005)

Speed.Dev.#1.....: 1832.0 MH/s (59.25ms)

Speed.Dev.#2.....: 1822.2 MH/s (59.57ms)

Speed.Dev.#3.....: 1824.8 MH/s (59.49ms)

Speed.Dev.#4.....: 1841.1 MH/s (58.96ms)

Speed.Dev.#5.....: 1696.2 MH/s (64.00ms)

Speed.Dev.#6.....: 1879.5 MH/s (57.75ms)

Speed.Dev.#7.....: 1749.0 MH/s (62.07ms)

Speed.Dev.#8.....: 1920.5 MH/s (56.52ms)

Speed.Dev.#*.....: 14565.2 MH/s

Hash type: MSSQL(2012)

Speed.Dev.#1.....: 180.1 MH/s (74.78ms)

Speed.Dev.#2.....: 180.1 MH/s (75.66ms)

Speed.Dev.#3.....: 180.7 MH/s (75.41ms)

Speed.Dev.#4.....: 181.1 MH/s (74.34ms)

Speed.Dev.#5.....: 180.7 MH/s (74.50ms)

Speed.Dev.#6.....: 180.3 MH/s (75.55ms)

Speed.Dev.#7.....: 180.2 MH/s (75.60ms)

Speed.Dev.#8.....: 180.3 MH/s (75.58ms)

Speed.Dev.#*.....: 1443.5 MH/s

Hash type: MySQL323

Speed.Dev.#1.....: 18603.4 MH/s (93.75ms)

Speed.Dev.#2.....: 19034.4 MH/s (91.63ms)

Speed.Dev.#3.....: 19019.5 MH/s (91.71ms)

Speed.Dev.#4.....: 19411.8 MH/s (89.85ms)

Speed.Dev.#5.....: 19523.0 MH/s (89.34ms)

Speed.Dev.#6.....: 19824.0 MH/s (87.98ms)

Speed.Dev.#7.....: 18995.3 MH/s (91.82ms)

Speed.Dev.#8.....: 19696.5 MH/s (88.54ms)

Speed.Dev.#*.....: 154.1 GH/s

Hash type: MySQL4.1/MySQL5

Speed.Dev.#1.....: 871.1 MH/s (62.55ms)

Speed.Dev.#2.....: 863.1 MH/s (63.14ms)

Speed.Dev.#3.....: 892.0 MH/s (61.07ms)

Speed.Dev.#4.....: 921.5 MH/s (59.10ms)

Speed.Dev.#5.....: 834.7 MH/s (65.28ms)

Speed.Dev.#6.....: 894.4 MH/s (60.91ms)

Speed.Dev.#7.....: 815.6 MH/s (66.81ms)

Speed.Dev.#8.....: 905.8 MH/s (60.15ms)

Speed.Dev.#*.....: 6998.2 MH/s

Hash type: md5apr1, MD5(APR), Apache MD5

Speed.Dev.#1.....: 2497.0 kH/s (84.13ms)

Speed.Dev.#2.....: 2533.6 kH/s (82.91ms)

Speed.Dev.#3.....: 2510.8 kH/s (83.65ms)

Speed.Dev.#4.....: 2672.9 kH/s (78.54ms)

Speed.Dev.#5.....: 2418.1 kH/s (86.89ms)

Speed.Dev.#6.....: 2667.8 kH/s (78.70ms)

Speed.Dev.#7.....: 2491.5 kH/s (84.30ms)

Speed.Dev.#8.....: 2681.7 kH/s (78.29ms)

Speed.Dev.#*.....: 20473.3 kH/s

Hash type: SHA-1(Base64), NSS LDAP, Netscape LDAP SHA

Speed.Dev.#1.....: 1911.3 MH/s (56.79ms)

Speed.Dev.#2.....: 1973.7 MH/s (54.99ms)

Speed.Dev.#3.....: 1907.3 MH/s (56.89ms)

Speed.Dev.#4.....: 2015.0 MH/s (53.86ms)

Speed.Dev.#5.....: 1884.5 MH/s (57.60ms)

Speed.Dev.#6.....: 2009.2 MH/s (54.01ms)

Speed.Dev.#7.....: 1853.6 MH/s (58.54ms)

Speed.Dev.#8.....: 2017.0 MH/s (53.81ms)

Speed.Dev.#*.....: 15571.6 MH/s

Hash type: SSHA-1(Base64), nsldaps, Netscape LDAP SSHA

Speed.Dev.#1.....: 1961.5 MH/s (55.34ms)

Speed.Dev.#2.....: 1940.2 MH/s (55.95ms)

Speed.Dev.#3.....: 1892.6 MH/s (57.34ms)

Speed.Dev.#4.....: 2031.0 MH/s (53.44ms)

Speed.Dev.#5.....: 1883.5 MH/s (57.63ms)

Speed.Dev.#6.....: 2011.3 MH/s (53.96ms)

Speed.Dev.#7.....: 1863.8 MH/s (58.23ms)

Speed.Dev.#8.....: 2029.2 MH/s (53.48ms)

Speed.Dev.#*.....: 15613.0 MH/s

Hash type: SSHA-512(Base64), LDAP {SSHA512}

Speed.Dev.#1.....: 177.3 MH/s (76.82ms)

Speed.Dev.#2.....: 177.6 MH/s (76.70ms)

Speed.Dev.#3.....: 178.4 MH/s (76.33ms)

Speed.Dev.#4.....: 178.7 MH/s (76.25ms)

Speed.Dev.#5.....: 177.4 MH/s (76.83ms)

Speed.Dev.#6.....: 178.2 MH/s (76.45ms)

Speed.Dev.#7.....: 177.4 MH/s (76.80ms)

Speed.Dev.#8.....: 177.9 MH/s (76.56ms)

Speed.Dev.#*.....: 1423.0 MH/s

Hash type: LM

Speed.Dev.#1.....: 3758.6 MH/s (57.87ms)

Speed.Dev.#2.....: 3751.9 MH/s (58.01ms)

Speed.Dev.#3.....: 3803.6 MH/s (57.19ms)

Speed.Dev.#4.....: 3772.9 MH/s (57.65ms)

Speed.Dev.#5.....: 3763.3 MH/s (57.82ms)

Speed.Dev.#6.....: 3796.4 MH/s (57.30ms)

Speed.Dev.#7.....: 3761.3 MH/s (57.83ms)

Speed.Dev.#8.....: 3775.4 MH/s (57.65ms)

Speed.Dev.#*.....: 30183.4 MH/s

Hash type: NTLM

Speed.Dev.#1.....: 8089.8 MH/s (53.88ms)

Speed.Dev.#2.....: 8196.5 MH/s (53.18ms)

Speed.Dev.#3.....: 8123.2 MH/s (53.66ms)

Speed.Dev.#4.....: 8549.1 MH/s (50.99ms)

Speed.Dev.#5.....: 7940.4 MH/s (54.90ms)

Speed.Dev.#6.....: 8424.6 MH/s (51.74ms)

Speed.Dev.#7.....: 7883.6 MH/s (55.29ms)

Speed.Dev.#8.....: 8519.3 MH/s (51.16ms)

Speed.Dev.#*.....: 65726.4 MH/s

Hash type: MS-AzureSync PBKDF2-HMAC-SHA256

Speed.Dev.#1.....: 2724.5 kH/s (74.28ms)

Speed.Dev.#2.....: 2810.0 kH/s (71.90ms)

Speed.Dev.#3.....: 2750.2 kH/s (73.44ms)

Speed.Dev.#4.....: 2882.1 kH/s (69.83ms)

Speed.Dev.#5.....: 2704.5 kH/s (74.85ms)

Speed.Dev.#6.....: 2882.5 kH/s (69.99ms)

Speed.Dev.#7.....: 2669.8 kH/s (75.84ms)

Speed.Dev.#8.....: 2869.0 kH/s (70.31ms)

Speed.Dev.#*.....: 22292.8 kH/s

Hash type: descrypt, DES(Unix), Traditional DES

Speed.Dev.#1.....:   176.1 MH/s (77.25ms)

Speed.Dev.#2.....:   175.5 MH/s (77.50ms)

Speed.Dev.#3.....:   176.1 MH/s (77.24ms)

Speed.Dev.#4.....:   175.9 MH/s (77.27ms)

Speed.Dev.#5.....:   175.3 MH/s (77.59ms)

Speed.Dev.#6.....:   176.8 MH/s (76.91ms)

Speed.Dev.#7.....:   175.0 MH/s (77.71ms)

Speed.Dev.#8.....:   175.8 MH/s (77.34ms)

Speed.Dev.#*.....: 1406.5 MH/s

Hash type: BSDiCrypt, Extended DES

Speed.Dev.#1.....: 519.8 kH/s (67.84ms)

Speed.Dev.#2.....: 524.3 kH/s (67.25ms)

Speed.Dev.#3.....: 524.9 kH/s (67.17ms)

Speed.Dev.#4.....: 522.2 kH/s (67.52ms)

Speed.Dev.#5.....: 516.6 kH/s (68.26ms)

Speed.Dev.#6.....: 525.7 kH/s (67.07ms)

Speed.Dev.#7.....: 503.2 kH/s (70.09ms)

Speed.Dev.#8.....: 524.2 kH/s (67.27ms)

Speed.Dev.#*.....: 4161.0 kH/s

Hash type: md5crypt, MD5(Unix), FreeBSD MD5, Cisco-IOS MD5

Speed.Dev.#1.....: 2660.3 kH/s (59.16ms)

Speed.Dev.#2.....: 2578.8 kH/s (61.05ms)

Speed.Dev.#3.....: 2601.5 kH/s (60.52ms)

Speed.Dev.#4.....: 2650.6 kH/s (59.39ms)

Speed.Dev.#5.....: 2469.2 kH/s (63.78ms)

Speed.Dev.#6.....: 2608.1 kH/s (60.35ms)

Speed.Dev.#7.....: 2453.8 kH/s (64.19ms)

Speed.Dev.#8.....: 2624.1 kH/s (60.00ms)

Speed.Dev.#*.....: 20646.5 kH/s

Hash type: bcrypt, Blowfish(OpenBSD)

Speed.Dev.#1.....: 1796 H/s (26.05ms)

Speed.Dev.#2.....: 1789 H/s (26.16ms)

Speed.Dev.#3.....: 1817 H/s (25.75ms)

Speed.Dev.#4.....: 1794 H/s (26.08ms)

Speed.Dev.#5.....: 1814 H/s (25.79ms)

Speed.Dev.#6.....: 1815 H/s (25.79ms)

Speed.Dev.#7.....: 1802 H/s (25.97ms)

Speed.Dev.#8.....: 1812 H/s (25.83ms)

Speed.Dev.#*.....: 14439 H/s

Hash type: sha256crypt, SHA256(Unix)

Speed.Dev.#1.....: 106.0 kH/s (50.66ms)

Speed.Dev.#2.....: 105.3 kH/s (51.00ms)

Speed.Dev.#3.....: 105.5 kH/s (50.86ms)

Speed.Dev.#4.....: 108.5 kH/s (49.48ms)

Speed.Dev.#5.....: 102.5 kH/s (52.35ms)

Speed.Dev.#6.....: 108.0 kH/s (49.73ms)

Speed.Dev.#7.....: 103.4 kH/s (51.89ms)

Speed.Dev.#8.....: 109.1 kH/s (49.22ms)

Speed.Dev.#*.....: 848.3 kH/s

Hash type: sha512crypt, SHA512(Unix)

Speed.Dev.#1.....: 35327 H/s (76.44ms)

Speed.Dev.#2.....: 33959 H/s (79.48ms)

Speed.Dev.#3.....: 34833 H/s (77.47ms)

Speed.Dev.#4.....: 35590 H/s (75.89ms)

Speed.Dev.#5.....: 33576 H/s (80.39ms)

Speed.Dev.#6.....: 34901 H/s (77.33ms)

Speed.Dev.#7.....: 34829 H/s (77.48ms)

Speed.Dev.#8.....: 35363 H/s (76.38ms)

Speed.Dev.#*.....: 278.4 kH/s

Speed.Dev.#7.....: 689.5 MH/s (79.05ms)

Speed.Dev.#8.....: 743.6 MH/s (73.29ms)

Speed.Dev.#*.....: 5727.3 MH/s

CPU Test Machine Benchmark Results:

root@ip-172-31-43-196 hashcat-3.30]# ./hashcat64.bin -b

hashcat (v3.30) starting in benchmark mode...

OpenCL Platform #1: Intel(R) Corporation

==========

Device #1:Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz, 2047/14622 MB allocatable, 8MCU .

Hash type: MD4

Speed.Dev.#1.....: 391.4 MH/s (21.37ms)

Hash type: MD5

Speed.Dev.#1.....: 219.2 MH/s (38.22ms)

Hash type: Half MD5

Speed.Dev.#1.....: 143.6 MH/s (58.35ms)

Hash type: SHA1

Speed.Dev.#1.....: 120.4 MH/s (69.60ms)

Hash type: SHA256

Speed.Dev.#1.....: 48370.0 kH/s (86.66ms)

Hash type: SHA384

Speed.Dev.#1.....: 12959.8 kH/s (80.85ms)

Hash type: SHA512

Speed.Dev.#1.....: 12833.2 kH/s (81.65ms)

Hash type: SHA-3(Keccak)

Speed.Dev.#1.....: 11684.9 kH/s (89.68ms)

Hash type: SipHash

Speed.Dev.#1.....: 244.8 MH/s (34.21ms)

Hash type: RipeMD160

Speed.Dev.#1.....: 57361.1 kH/s (73.06ms)

Hash type: Whirlpool

Speed.Dev.#1.....: 3201.5 kH/s (81.50ms)

Hash type: GOST R 34.11-94

Speed.Dev.#1.....: 3857.5 kH/s (67.63ms)

Hash type: GOST R 34.11-2012 (Streebog) 256-bit

Speed.Dev.#1.....: 1378.2 kH/s (95.05ms)

Hash type: GOST R 34.11-2012 (Streebog) 512-bit

Speed.Dev.#1.....: 1378.1 kH/s (94.96ms)

Hash type: DES (PT = $salt, key = $pass)

Speed.Dev.#1.....: 25573.2 kH/s (81.79ms)

Hash type: 3DES (PT = $salt, key = $pass)

Speed.Dev.#1.....: 6376.6 kH/s (82.14ms)

Hash type: phpass, MD5(Wordpress), MD5(phpBB3), MD5(Joomla)

Speed.Dev.#1.....:   74729 H/s (54.44ms)

Hash type: scrypt

Speed.Dev.#1.....:   0 H/s (2.43ms)

Hash type: PBKDF2-HMAC-MD5

Speed.Dev.#1.....:   75724 H/s (53.31ms)

Hash type: PBKDF2-HMAC-SHA1

Speed.Dev.#1.....:   45658 H/s (88.66ms)

Hash type: PBKDF2-HMAC-SHA256

Speed.Dev.#1.....:   17934 H/s (37.64ms)

Hash type: PBKDF2-HMAC-SHA512

Speed.Dev.#1.....: 6482 H/s (78.62ms)

Hash type: Skype

Speed.Dev.#1.....: 154.2 MH/s (54.36ms)

Hash type: WPA/WPA2

Speed.Dev.#1.....: 5624 H/s (90.53ms)

Hash type: IKE-PSK MD5

Speed.Dev.#1.....: 18647.3 kH/s (56.17ms)

Hash type: IKE-PSK SHA1

Speed.Dev.#1.....: 11367.2 kH/s (92.18ms)

Hash type: NetNTLMv1-VANILLA / NetNTLMv1+ESS

Speed.Dev.#1.....: 238.0 MH/s (35.19ms)

Hash type: NetNTLMv2

Speed.Dev.#1.....: 17148.7 kH/s (61.09ms)

Hash type: IPMI2 RAKP HMAC-SHA1

Speed.Dev.#1.....: 22893.4 kH/s (91.55ms)

Hash type: Kerberos 5 AS-REQ Pre-Authetype 23

Speed.Dev.#1.....: 3234.9 kH/s (80.00ms)

Hash type: Kerberos 5 TGS-REP etype 23

Speed.Dev.#1.....: 3306.6 kH/s (78.26ms)

Hash type: DNSSEC (NSEC3)

Speed.Dev.#1.....: 43621.6 kH/s (96.09ms)

Hash type: SHA-1(Base64), nsldap, Netscape LDAP SHA

Speed.Dev.#1.....: 120.2 MH/s (69.75ms)

Hash type: SSHA-1(Base64), nsldaps, Netscape LDAP SSHA

Speed.Dev.#1.....: 120.3 MH/s (69.70ms)

Hash type: SSHA-512(Base64), LDAP {SSHA512}

Speed.Dev.#1.....: 12853.2 kH/s (81.52ms)

Hash type: LM

Speed.Dev.#1.....: 25799.0 kH/s (81.07ms)

Hash type: NTLM

Speed.Dev.#1.....: 402.1 MH/s (20.81ms)

Hash type: Domain Cached Credentials (DCC), MS-Cache

Speed.Dev.#1.....: 125.0 MH/s (67.04ms)

Hash type: Domain Cached Credentials 2 (DCC2), MS-Cache 2

Speed.Dev.#1.....: 4526 H/s (90.33ms)

Hash type: MS-AzureSync PBKDF2-HMAC-SHA256

Speed.Dev.#1.....: 50307 H/s (52.63ms)

Hash type: descrypt, DES(Unix), Traditional DES

Speed.Dev.#1.....: 1055.6 kH/s (496.44ms)

Hash type: BSDiCrypt, Extended DES

Speed.Dev.#1.....: 15896 H/s (85.55ms)

Hash type: md5crypt, MD5(Unix), FreeBSD MD5, Cisco-IOS MD5

Speed.Dev.#1.....: 35248 H/s (57.66ms)

Hash type: bcrypt, Blowfish(OpenBSD)

Speed.Dev.#1.....: 2448 H/s (51.08ms)

Hash type: sha256crypt, SHA256(Unix)

Speed.Dev.#1.....: 1805 H/s (56.40ms)

Hash type: sha512crypt, SHA512(Unix)

Speed.Dev.#1.....: 1702 H/s (59.90ms)

Speed.Dev.#1.....: 0 H/s (39.28ms)

Hash rate denominations

1 kH/s is 1,000 (one thousand) hashes per second

1 MH/s is 1,000,000 (one million) hashes per second.

1 GH/s is 1,000,000,000 (one billion) hashes per second.

1 TH/s is 1,000,000,000,000 (one trillion) hashes per second.

1 PH/s is 1,000,000,000,000,000 (one quadrillion) hashes per second.

1 EH/s is 1,000,000,000,000,000,000 (one quintillion) hashes per second.

Conversions

1 MH/s = 1,000 kH/s

1 GH/s = 1,000 MH/s = 1,000,000 kH/s

1 TH/s = 1,000 GH/s = 1,000,000 MH/s = 1,000,000,000 kH/s

3.7 Testing Environment Diagram

The testing Environment of this research is shown below:

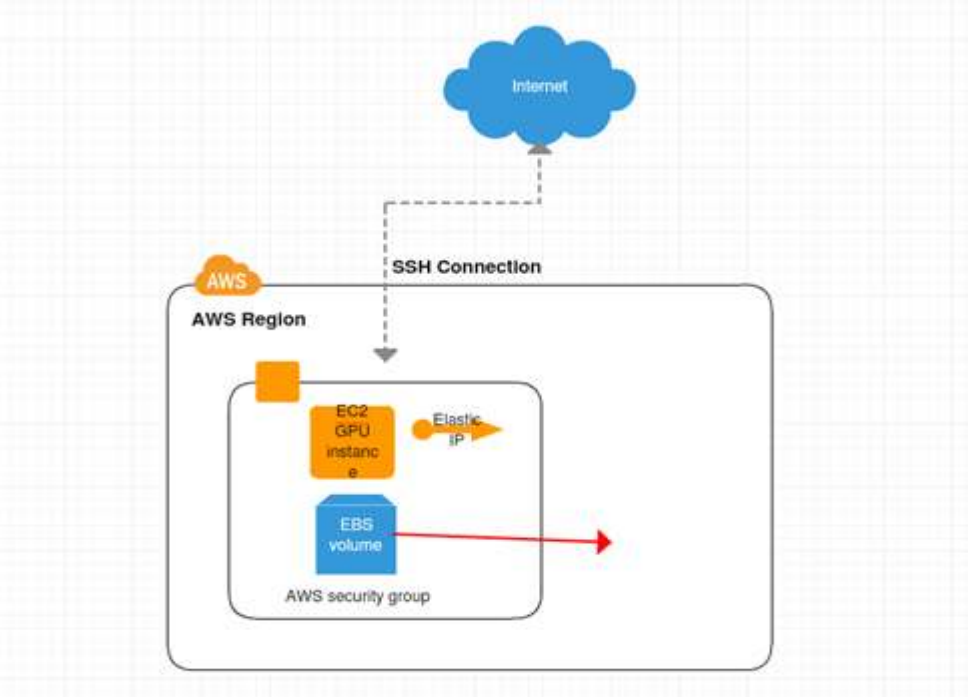


Figure 7. Testing Environment

3.8 Wordlist Selected

For the test, different filters and combination of uppercase letters, lowercase letters,digits, special characters will be used. The test will be done using thehybrid attack at first before applying brute force attacks where every combination of the characters will be tried.

3.9 Sample Password Hashed Dump File

In order to conduct the test some common password upto 8 characters in length characters' length and will hash it with MD5 and SHA1 as well as more cryptographically strong hashing algorithm bcrypt. The following sample password hash has been selected

Sample password list:

Password

HELLOO

MYSECRET

test1234

password!

You9can!

./?';,<>

Mysecret

TheMD5 and SHA1 hash of the generated password from the above is generated by the following code:

```
import hashlib
s='password'
sb=s.encode("utf8")
#MD5 generator
print (hashlib.md5(sb).hexdigest())
5f4dcc3b5aa765d61d8327deb882cf99
#SHA256 generator
print (hashlib.sha256(sb).hexdigest())
5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
```

MD5 Hash:

dc647eb65e6711e155375218212b3964

16454bd041c46012e31778eb94b8111a

958152288f2d2303ae045cffc43a02cd

16d7a4fca7442dda3ad93c9a726597e4

49f24c0c152b2375431210f9443d176f

b64b0e1165a77bd90a1673469f1af0e1

7185ad7f0851780a2db24edc8347b12a

06c219e5bc8378f3a8a3f83b4b7e4649

SHA256 Hash:

e7cf3ef4f17c3999a94f2c6f612e8a888e5b1026878e4e19398b23bd38ec221a

2d32d2db26a9a8e8b3a69a5739a17981a5a064a5c3037a8d3891ab3e41f57246

3fcdbc4a0ed38df8d4bd234e2c8ad3b2623fa5265f31763d1e91a848471a8a9b

937e8d5fbb48bd4949536cd65b8d35c426b80d2f830c5c308e2cdec422ae2244

c075349b9b6f6b3e41b34e4e71ac22a685102b0b2246c5f84d67c5eed3ad39fb

4824033be89e919a06ac33255b06761f706edc1ac8fc37b86574892ab7c3248d

9070906f306d5d34c301b9f4cda9f71c2a19543ccea44b6b08c18b3d76941936

652c7dc687d98c9889304ed2e408c74b611e86a40caa51c4b43f1dd5913c5cd0

We also generateabcrypt hash of the following password with the following code.

Sample Password:

HELLO

1!Su

Pass

pass

1234

:/?<

```
import bcrypt
password = b"Pass"
# Hash a password for the first time, with a certain number of rounds
hashed = bcrypt.hashpw(password, bcrypt.gensalt(14))
print(hashed)
b'$2b$14$4tWHOXsyYtuVD8CbzjMUbeNqfMKGKiECODTkQ4Zcf11wJ6nJAD7XW'
```

Here the bcrypt hash of sample password “Pass” was created with 14 round (2^14=16384) of generation via gensalt() function. For the sake of our test, we will be keeping it to minimal, less than 10.

The Resulting generated bcrypt hash of the above sample password was:

$2y$10$vHvY3cA252u/68KesxmOg.WIjkWOHQcFqXc.KRSV4aLn/pIC1D5ZC

$2y$10$eVaCxfaaMDXKS5LTu1uX2O6k3/lUpGE83luvxfoGdkM6HAMVWirvW

$2y$10$K8nQaEFpiZkSdkjKlXfjveu44pTKD/lvpOU2Cu/8INh2vPDUgS29e

$2y$10$wacXM0HGg/26pzRGvIEE4OMg4jGIHjhptHKMdowmdr4zpTyu0triC

$2y$10$FwaS9uiu7mkCIJ.d9fCYI.PsF8qY5I1ZdrbJ0qBBcHdDLhours3gxPLq

$2y$10$kjGTFQVsc7eislYriw0ibu1GZi1rUod0unnpY0rT9eeSIYGBJRiui

Chapter 4: Analyzing Results

4.1 Introduction

In this section, we will analyze the result performed in the previous test scenario and all the test results are included in the appendix section at the end of this paper. We end this section with recommendations based on analyzing the test result.

4.2 Test Results

We ran the CPU crack and GPU cracking on the sample MD5, SHA-256, and Bcrypt hash dump. We applied thedifferent filter as follows:

u= uppercase letters only – total 26 characters

l= lowercase letter only – total 26 characters

ul= uppercase and lowercase – total 52 characters

d= digits only – total 10 digits

s= special characters only – total 33 characters

ls/us= lowercase/uppercase with special characters – total 59 characters

usld= lowercase, uppercase, special character, and digits- total 95 characters

We also conducted an experimental run where we applied fixed characters in certain positions to observe any improvements in timing. All our CPU/GPU/Experimental test results are shown below in the following tables.

Table 4

MD5 crackingperformance With CPU machine

MD5 CPUWith312MH/s										
Filter	u	l	ul	ld	ls	s	ulsd			
HELLO (6u)	2sec									
tes1234 (7ld)				3min						
Passwor! (7ls)					1hours 13min					
Password (8ul)			1hours 12min							
mysecret (8l)		12min								
./?!';<> (8s)						1hours12min				
MYSECRET (8u)	12min									
You9can! (8ulsd)							*			
Full Round(8char)	12 min	12 min	1hours12min	2hours30min	5days1hours	1hours 25min	246days (estimated)			

In the Table -4 as the result is as follows:

- Using uppercase only character (u) it takes 2sec for 6-character long password, HELLO to crack where 8-character long password ‘MYSECRET’ takes 12 min and it finishes checking all the combination of 8 characters roughly at thesame time too.
- Using lowercase only character (l) it takes 12 min to crack 8-character long password ‘mysecret’ as well as finishes checking all the combination of 8 characters roughly at thesame time too.
- Using acombination of uppercase and lowercase character (ul) it takes 1 hour 12 min to finish cracking 8 char long ‘Password’ as well as finishes checking all the combination of 8 characters roughly in thesame time too.
- Using lowercase and digits (ld) it takes 3 min to crack 7 char long ‘tes1234’ password while it takes 2hours 30 min to finish checking all the combination ofthe 8 char field.
- Using lowercase and aspecialcharacter(ls) it takes it takes 1hours 13 min to find the 7-char password ‘Passwor!’ while it will take approximately 5days and 1hours to finish checking all the combination.

  The way we estimated 8 char long password cracking time with lowercase and special characters are as follows:

  - ✓ Total characters: lowercase (26) and special characters (33) =33+26=59
  - ✓ Total combination possible 59^8=1.4683044e+14
  - ✓ MD5 CPU cracking speed= 312MHS = 312000000H/s

- ✓ Cracking time in days= 1.4683044e+14/312000000=470610.376937s/3600=130.725104705hours/24 =5days 1hours

- Using the special characters only it takes 1hours 12 min to finish cracking 8 char long password ‘./?!’;<>’ while it takes 1hours25min to finish checking all the special character combination of the8char field.
- At last we try all the lowercase, uppercase, special characters and digits(usld) for all the 8 filed of the password and though we could not find out 8 char long password ‘You9can!’ we did find to estimate how long it will take to look up all the (USLD) combination of each 8-char filed with our CPU machine with thefollowing calculation:
  - ✓ Total characters: lowercase (26) and special characters (33), uppercase (26) and digits (10) =33+26+26+10=95
  - ✓ Total combination possible 〖95〗^8= 6.6342043e+15
  - ✓ MD5 CPU cracking speed= 312MHS = 312000000H/s
  - ✓ Cracking time in days= 6.6342043e+15/ 312000000H/s =21263475.3618s/3600=5906.52093384hours/24=246days

Table 5

MD5 cracking performance With GPU machine

MD5 GPUWith21117MH/s							
Filter	u	l	ul	ld	ls	s	ulsd
HELLO (6u)	1sec						10 sec
tes1234 (7ld)				2sec			20min
Passwor! (7ls)					2min		
Password (8ul)			12min				
mysecret (8l)		10sec					
./?!';<> (8s)						7sec	1hours
MYSECRET (8u)	10sec						1hours
You9can! (8ulsd)							1hours
Full Round(8char)	30 sec	30sec	12min	2min	2hours	1min28sec	1hours(7char)4days (estimated)

In the Table -5 as the result is as follows:

- Using uppercase only character (u) it takes 1sec from 6-character long password, HELLO to crack where 8-character long password ‘MYSECRET’ takes 10 Sec and it finishes checking all the combination of 8 characters roughly in 30 secs.
- Using lowercase only character (l) its takes 10 secs to crack 8-character long password ‘mysecret’ as well as finishes checking all the combination of 8 characters roughly in 30 secs.
- Using acombination of uppercase and lowercase character (ul) it takes 12 min to finish cracking 8 char long ‘Password’ as well as finishes checking all the combination of 8 characters roughly in thesame time too.
- Using lowercase and digits (ld) it takes 2secs to crack 7 char long ‘tes1234’ password while it takes 2 mins to finish checking all the combination of the 8 character field.
- Using lowercase and aspecialcharacter(ls) it takes it takes 2 min to find the 7-char password ‘Passwor!’ while it will take approximately 2hours to finish checking all the combination.
- Using the special characters only it takes 7sec to finish cracking 8 char long password ‘./?!’;<>’ while it takes 1min 28 sec to finish checking all the special character combination of the8char field.
- At last, we try all the lowercase, uppercase, special characters and digits(usld) for all the 8 field of the password and for the 8-char long password (. /?!’;<>,MYSECRET,You9can!) . We could not finish the all the combination of 8 chars but we estimated it will take around 4 days to finish checking all the possible combination

of 8 -char password using our GPU machine. The way we estimated the cracking time as follows:

- ✓ Total characters: lowercase (26) and special characters (33), uppercase (26) and digits (10) =33+26+26+10=95
- ✓ Total combination possible 〖95〗^8= 6.6342043e+15
- ✓ MD5 GPU cracking speed= 21117MHS = 21117000000H/s
- ✓ Cracking time in days= 6.6342043e+15/ 21117000000H/s =314164.14798s/3600=87.2678188833hours/24=4days

Table 6

SHA-256 cracking performance With CPU machine

SHA-256 CPU with 88954KH/s							
Filter	u	l	ul	ld	ls	s	ulsd
HELLO (6u)	15sec		34sec				1h 9 mins
tes1234 (7ld)				15min			9dys
Passwor! (7ls)					15min		9days
Password (8ul)			3hours				
mysecret (8l)		26min					
./?!';<> (8s)						20min	*
MYSECRET (8u)	20min						*
You9can! (8ulsd)							*
Full Round(8char)	38min	38min	7days 20 hours	9hours 25mins	7hours 50mins (7char)	9hours	9days(7char) 863days(8char) (estimated)

In the Table -6 as the result is as follows:

- Using uppercase only character (u) it takes 15sec for 6-character long password HELLO to crack where 8-character long password ‘MYSECRET’ takes 20 mins and it finishes checking all the combination of 8 characters roughly in 38mins.
- Using lowercase only character (l) it takes 26 min to crack 8-character long password ‘mysecret’ as well as finishes checking all the combination of 8 characters roughly in 38mins.
- Using a combination of uppercase and lowercase character (ul) it takes 3 hours to finish cracking 8 char long ‘Password’ as well as finishes checking all the combination of 8 characterswill take estimated 7days 20 hours to finish.

  The way we estimated the cracking time as follows:

  - ✓ Total characters: lowercase (26) , uppercase (26) =26+26=52
  - ✓ Total combination possible 〖52〗^8= 5.3459729e+13
  - ✓ SHA-256 CPU cracking speed,88954KH/s=88954000h/s
  - ✓ Cracking time in days= 5.3459729e+13/88954000=600981.726864s/3600=166.939368573hours/24= 7days.
- Using lowercase and digits (ld) it takes 15 min to crack 7 char long ‘tes1234’ password while it takes 9hours 25 min to finish checking all the combination ofthe 8 char field.
- Using lowercase and specialcharacter(ls) it takes it takes 15 min to find the 7-char password ‘Passwor!’ while it will take 7hours 50 min to check 7 charsfield.

- Using the special characters only it takes 20 min to finish cracking 8 char long passwords ‘./?!’;<>’ while it takes 9hours to finish checking all the special character combination of8char field.
- At last we try all the lowercase, uppercase, special characters and digits(usld) for all the 8 field of the password and though we could not find out 8 char long password ‘You9can!’ we did find to estimate how long it will take to look up all the (USLD) combination of each 7 char and 8-char filed with our CPU machine with following calculation

Time Estimation for 7-char password

- ✓ Total characters: lowercase (26) and special characters (33), uppercase (26) and digits (10) =33+26+26+10=95
- ✓ Total combination possible 〖95〗^7= 6.983373e+13
- ✓ SHA-256 CPU cracking speed, 88954KH/s=88954000h/s
- ✓ Cracking time in days= 6. 983373e+13/ 88954000h /s=785054.405753s/3600=218.070668265/24=9days

Time Estimation for 8-char password

- ✓ Total characters: lowercase (26) and special characters (33), uppercase (26) and digits (10) =33+26+26+10=95
- ✓ Total combination possible 〖95〗^8= 6.6342043e+15
- ✓ SHA-256 CPU cracking speed,88954KH/s=88954000h/s
- ✓ Cracking time in days= 6.6342043e+15/ 88954000h /s =74580168.5466 s/3600=20716.7134852 hours/24=863days

Table 7

SHA-256 cracking performance With GPU machine

SHA-256 GPUWith5325MH/s							
Filter	u	l	ul	ld	ls	s	ulsd
HELLO (6u)			1sec				2min
tes1234 (7ld)				1min			3hours
Passwor! (7ls)					8min		3hours
Password (8ul)			1min				
mysecret (8l)		1min		1min 22sec			
./?!';<> (8s)						2min	*
MYSECRET (8u)	1min						*
You9can! (8ulsd)							*
Full Round(8char)	1min 6sec	1min 6sec	2hours 57min	10min	8hours	5min	3hours 50min (7char) 14days10hours (8char)

In the Table -7 as the result is as follows:

- Using uppercase only character (u) to crack 8-character long password ‘MYSECRET’ takes 1 min and it finishes checking all the combination of 8 characters roughly in 1min 6secs.
- Using lowercase only character (l) it takes 1 min to crack 8-character long password ‘mysecret’ as well as finishes checking all the combination of 8 characters roughly in 1min 6secs.
- Using a combination of uppercase and lowercase character (ul) it takes 1 min to finish cracking 8 char long ‘Password’ as well as finishes checking all the combination of 8 characters roughly in 2hours 87min.
- Using lowercase and digits (ld) it takes 1min to crack 7 char long ‘tes1234’ password and 1 min 22sec for 8-char password ‘mysecret’, while it takes 10 min to finish checking all the combination ofthe 8 char field.
- Using lowercase and specialcharacter(ls) it takes it takes 8 min to find the 7-char password ‘Passwor!’ while it will take approximately 8hours to finish checking all the combination.
- Using the special characters only it takes 2 min to finish cracking 8 char long password ‘./?!’;<>’ while it takes 5min to finish checking all the special character combination of8char field.
- At last we try all the lowercase, uppercase, special characters and digits(usld). It took around 3 hours to finish checking 7 char length password while for all the 8 field of the password and for the 8-char long password (. /?!’;<>,MYSECRET,You9can!).Though

we could not finish 8 char password, but we estimated it will take around 10days and 10 hours to finish checking all the combination of the 8char field. The estimated time is calculated as follows:

- ✓ Time Estimation for 8-char password
- ✓ Total characters: lowercase (26) and special characters (33), uppercase (26) and digits (10) =33+26+26+10=95
- ✓ Total combination possible 〖95〗^8= 6.6342043e+15
- ✓ SHA-256 GPU cracking speed, 5325MH/s=5325000000h/s
- ✓ Cracking time in days= 6.6342043e+15/5325000000h/s= 1245859.96486/3600=346.0722hours/24=14days

Table 8

Bcrypt cracking performance With CPU machine

Bcrypt-CPU 90H/s							
ulsd	s	d	l	ul	u	Filter	
				*		Pass(4ul)	
			3hours 20min			Pass(4l)	
		9min				1234(4d)	
	3hours					./?<(4s)	
*						1!Su(4usld)	
					*	HELLO(5u)	
3char 15hours 4 mins 4char 10 days 10hours (estimated)	3char 36min 4char 4 hours	4char 9min	3char 2hours 4char 8 hours	3char 2hours 21min 4char 22hours 55mins (estimated)	3char 21min 4char 8hours	Full Round	

In the Table -8 as the result is as follows:

- Using uppercase only character (u) it takes 21min to go through all combinations of 3 char length, whereas it takes 8hours to finish 4 char long.
- Using uppercase and lowercase (ul) it takes 2 hours 21 min to finish 3 char length passwords. We could not finish cracking 4 char length passwords ‘Pass’ as we estimated it will take 3dys 17hours to finish checking all the combination. The estimation is calculated as follows:

  Time Estimation for 4-char password

  - ✓ Total characters: lowercase (26), uppercase (26) =52
  - ✓ Total combination possible 〖52〗^4= 7311616
  - ✓ Bcrypt CPU machine cracking speed=90h/s
  - ✓ Cracking time in days=

    7311616/90=81240.1777778s/3600=2031.004hr/24=22hours 50 mins
- Using lowercase only character (l) its takes 3hours 20 min to crack 4-character long password ‘Pass’ and to finish all the combination of 4 chars it takes 8hour where for 3 chars it takes 2hour.
- Using only digits(d) it takes 9min to crack 4 char password ‘1234’ as well as roughly going through all the combination of 4-char field.
- Using only special characters(s) it takes 36 mins to finish checking all the combination of 3 char length password whereas it found 4 char length password ‘,/?<(‘ in 3 hours while taking 4 hours to go through all the combination of the 4 char length password.

- At last we try all the lowercase, uppercase, special characters and digits(usld). It takes 15hours 4 min to finish going through all the combination of the 3-char length. We could not finish checking for our 4 char length password ‘1!Su’ as we estimated it will take 10 days 10hours to check all the possible combinations using following formulas
    - ✓ Time Estimation for 4-char password
    - ✓ Total characters: lowercase (26), uppercase (26), Special characters(33), digits(10) =95
    - ✓ Total combination possible 〖95〗^4= 81450625
    - ✓ Bcrypt CPU machine cracking speed=90h/s
    - ✓ Cracking time in days=81450625/90=905006.944444/3600s=251.390817901hr =10days

Table 9

Bcrypt cracking performance With GPU machine

Bcrypt-GPU 520H/s							
usld	s	d	l	ul	u	Filter	
						Pass(4ul)	
			1hours			Pass(4l)	
		6min				1234(4d)	
						./?<(4s)	
*	1hour 20 min					1!Su(4usld)	
					1hour 20 min	HELLO(5u)	
3 Char 2hours 40 min 4 char 1 day 19hours	4char 2hours	5char 10 min	4char 1hours 20 min	3char 23 min 4 char 4hours 30 mins	5 char 2hours 40 min 4char 1hours	Full Round	

In the Table -9 as the result is as follows:

- Using uppercase only character (u) it takes1hour to go through all combinations of4 char length, whereas it takes 2hours to finish 5 char length passwords. It found our 5-char length password ‘HELLO’ in 1 hour 20 mins.
- Using uppercase and lowercase (ul) it takes 35 mins to crack 4 char length password ‘Pass’ .To go through all the combination of 3 char length passwords it takes 23 mins while to 4 char length passwords it takes 4 hours 30 mins only.
- Using lowercase only character (l) it takes 1hour to crack 4-character long password ‘Pass’ and to finish all the combination of 4 chars it takes 1hour 20 mins only.
- Using only digits(d) it takes 6min to crack 4 char password ‘1234’. It finished checking all the combination of 5 char length passwords in about 10 mins.
- Using only special characters(s) it took 1 hour 20 mins to crack 4 char password ‘1!Su’ and going through all the combination in roughly about 2 hours.
- At last we try all the lowercase, uppercase, special characters and digits(usld). It takes 2hours 40 mins to finish going through all the combination of the 3-char length. We could not finish checking for our 4 char length password ‘1!Su’ as we estimated it will take 1 day 19hours to check all the possible combinations using following formulas
    - ✓ Time Estimation for 4-char password
    - ✓ Total characters: lowercase (26), uppercase (26), Special characters(33), digits(10) =95
    - ✓ Total combination possible 〖95〗^4= 81450625
    - ✓ Bcrypt CPU machine cracking speed=520h/s

- ✓ Cracking time in days=81450625/520=156635.817308 /3600s=43.5099492521 hr =1day 19hours.

4.3 Experimental Run

- MD5 GPU with 1st character set as uppercase (U) and last character set as special characters(s) while all other character is combination of lowercase (l), uppercase (u), special character(s), digits (d) takes only 2mins to finish whole 8-character set.
- MD5 GPU with last character fixed as special characters(s) and trying all other combination (lowercase (l), uppercase (u), special character(s), digits (d)) in first 7 character takes only 8hours to finish.
- MD5 GPU trying all first 7 char as lowercase(l) and special character(s) whole last character fixed as special character(s) makes the cracking time of 8-character set to only 1min
- SHA-256 GPU machine cracking 8 characters with combination of lowercase (l), uppercase (u), special character(s), digits (d) in 2nd to 7th character while making the 1st character fixed for special characters(s) and 8th character fixed for uppercase (u) brings the cracking time to only 6mins while just making the 1st character fixed for uppercase(s) letters makes the cracking time around 10hours.

4.4 Analyzing Result

- Using CPU instances with the combination of characters like uppercase, lowercase, special characters and digits a password length of 8 using MD5 hash takes 246 days to decrypt while using GPU it takes only 3 days.

- Similarly,the same password length using more secured SHA-256 hashing algorithm takes 863 days for our CPU machine to crack where with our GPU its only 10 days.
- Using more secured and computationally intensive Bcrypt hashing algorithm a password length of 4 characters only with a combination of characters like uppercase, lowercase, special characters and digits it takes our CPU instances 84 days to crack whereas with our GPU only 12 days.
- GPU instancing took only 5 min to crack SHA-256 password length of 8 with special characters only, whereas it took almost 3hours to crack password length of 4 with Bcrypt hashing algorithm

4.5 Recommendation

- Use password random generator to make a strong Radom password. One such sample random password generator script is given below:

  Alphanumeric Password Generator Script

```
import random

import string

str = []

chars = string.ascii_letters + string.punctuation  + string.digits

num = int(input('How long do you want the string to be?  '))

for k in range(1, num+1):

 str.append(random.choice(chars))

str = "".join(str)

print (str)
```

- Never reuse the same password for different accounts.
- As a security administrator or developer try to use a modern hashing algorithm like Bcrypt which is slow in computing using GPU or CPU.
- Never store password without hashing.
- Always add salt to the hashed password for added security.
- Password length should be at least 10 characters in length and use combination of characters like uppercase, lowercase, special characters and digits and never use dictionary words.
- Avoid using words from dictionary which can be easily brute-force by dictionary attack.

Chapter 5: Conclusion

5.1 Introduction

In the last section of our paper, we provide the timeline of our project as well as future work direction and closing remarks of our paper.

5.2 Timeline

Here is the timeline of my tasks that will be undertakenthoursough my research:

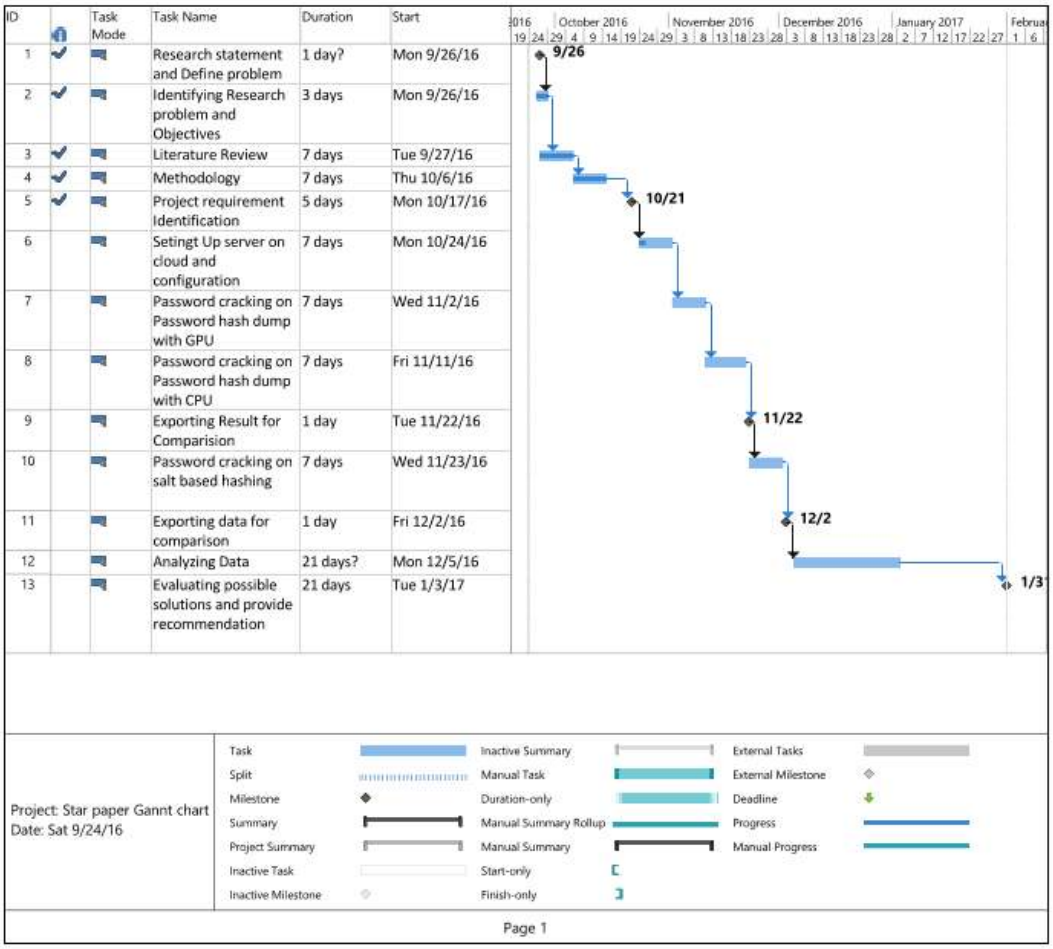


Figure 8. Project Timeline

5.3 Future Work

In the future, we plan to test whether adding salt (random number) with MD5 and SHA-256 increase the cracking time it takes with GPU or not. Also we didn't test the effect of fixed certain character types in the password field with Bcrypt hashing algorithm like we did in section 3.11 with SHA-256 and MD5 hash. So, we would like to give it a try with Bcrypt and compare results with SHA-256 and MD5.

5.4 Conclusion

In the end the aim of our paper was to compare the effectiveness of a GPU based, password cracking over the CPU as well the weakness in contemporary password hashing algorithm used (SHA-256, MD5) in today and why we should use a modern hashing algorithm like Bcrypt over SHA-256 and MD5 and why should use more complex passwords. We also came into conclusion that using salt with a hashed password add more computational cost to crack even with Highly capable GPU. We also presented a test bed scenario where a normal user can leverage the power of cloud computing to crack relatively complex password relatively easily.

REFERENCES

1. Lawrence O'Gorman, "Comparing Passwords, Tokens, and Biometrics for User Authentication," Proceedings of the IEEE, vol. 91, no. 12, pp. 2021 - 2040, 2003.

2. Anne Adams, Martina Angela Sasse, and Peter Lunt, "Making Passwords Secure and Usable," Proceedings of HCI on People and Computers, vol. XII, pp. 1 - 19, 1997.

3. MD5 Message Digest Algorithm Hash Collision Weakness. (2016). Securityfocus.com. Retrieved 24 September 2016, from http://www.securityfocus.com/bid/11849/discuss

4. Qiu, W., Gong, Z., Guo, Y., Liu, B., Tang, X., & Yuan, Y. (2016). GPU-Based High-Performance Password Recovery Technique for Hash Functions. ResearchGate. Retrieved 24 September 2016, from https://www.researchgate.net/publication/292761539_GPUBased_High_Performance_Password_Recovery_Technique_for_Hash_Functions

5. About Secure Password Hashing « Stack Exchange Security Blog. (2016). Security.blogoverflow.com. Retrieved 24 September 2016, from http://security.blogoverflow.com/2013/09/about-secure-password-hashing/

6. Fritz Bauspiess , Frank Damm, Requirements for Cryptographic hash functions, Computers, and Security, v.11 n.5, p.427-437, Sept. 1992 [doi>10.1016/0167-4048(92)90007-E]

7. Anon, (2016). Computing.dcu.ie. Retrieved 25 September 2016, from http://www.computing.dcu.ie/~hamilton/teaching/CA642/notes/Hash.pdf

8. Robin Thomas Jose, and C.G Thomas 2320-9798 (Print): A Comparative Study on Different Hashing Algorithms 3.7 (2015): n.7 p-1-6 International Journal of Innovative Research in Computer and Communication Engineering, 7 Aug. 2015. Web. 24 Sept. 2016.

9. A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of Applied Cryptography. CRC Press, fifth edition, October 1996.

10. A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. Handbook of applied cryptography. CRC, 1997. [cited at p. 8, 14, 15]

11. D. Reid and C. Knipping, Proof in mathematics education: Research, learning and teaching.2010

12. D. Florencio and C. Herley, “A large-scale study of web password habits,” 2007.

13. R. Shirey, “RFC2828 Internet security glossary.” Retrieved 25 September 2016, Fromhttps://tools.ietf.org/html/rfc2828

14. Hellman, M. 1980. A cryptanalytic time-memory tradeoff. IEEE Trans. Inf. Theor. 26, 401—406

15. Password Strength Checker. (2016). Passwordmeter.com. Retrieved 26 September 2016, from http://www.passwordmeter.com/

16. Cognitive Disabilities and the Web: Where Accessibility and Usability Meet. (2016). Ncdae.org. Retrieved 26 September 2016, from http://ncdae.org/resources/articles/cognitive/

17. Yulong Yang, JanneLindqvist, and Antti Oulasvirta. 2014. Text Entry Method Affects Password Security. Computing Research Repository (2014). http://arxiv.org/abs/1403.1910

18. Karl Rupp. Retrieved September 25, 2016,”CPU, GPU and MIC Hardware Characteristics over Time,” from https://www.karlrupp.net/2013/06/cpu-gpu-and-mic-hardware-characteristics-over-time/

19. Y. Liu E. Wu. Emerging technology about GP-GPU. In Circuits and Systems- Asia Pacific Conference, pages 618{622. IEEE, Dec 2008

20. D. Geer. Taking the graphics processor beyond graphics. Computer, 38:14{16, Sep 2005.

21. T. R. Halfhill, “Parallel Processing with CUDA,”Microprocessor Report, January 28, 2008

22. P. Mell and T. Grance, The NIST Definition of Cloud Computing (Special Publication 800-145), 2011.

23. C. Vecchiola, S. Pandey and R. Buyya, "High-Performance Cloud Computing: A View of Scientific Applications."

24. C. J. Thompson, S. Hahn, and M. Oskin. Using modern graphics architectures for general-purpose computing: A framework and analysis. In Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture, pages 306{317. IEEE Computer Society Press, 2002. [cited at p. 2]

25. D. L. Cook, J. Ioannidis, A. D. Keromytis, and J. Luck. CryptoGraphics: Secret key cryptography using graphics cards. Topics in Cryptology {CT-RSA 2005, pages 334{350, 2005. [cited at p. 2]

26. J. Yang and J. Goodman. Symmetric key cryptography on modern graphics hardware. Advances in Cryptology {ASIACRYPT 2007, pages 249{264, 2008. [cited at p. 2,35]

27. A. Di Biagio, A. Barenghi, G. Agosta, and G. Pelosi. The design of a parallel AES for graphics hardware using the CUDA framework. In Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on, pages 1{8. IEEE, 2009. [cited at p. 3]

28. S. A. Manavski. CUDA compatible GPU as an efficient hardware accelerator forAES cryptography. In Signal Processing and Communications, 2007. ICSPC 2007. IEEE International Conference on, pages 65{68. IEEE, 2008. [cited at p. 3, 85]

29. O. Harrison and J. Waldron. AES encryption implementation and analysis on commodity graphics processing units. Cryptographic Hardware and Embedded SystemsCHES 2007, pages 209{226, 2007. [cited at p. 3, 85]

30. D. Bernstein, H. C. Chen, C. M. Cheng, T. Lange, R. Niederhagen, P. Schwabe, and B. Y. Yang. ECC2K-130 on NVIDIA GPUs. Progress in Cryptology-INDOCRYPT 2010, pages 328{346, 2010. [cited at p. 3, 85]

31. G. Hu, J. Ma, and B. Huang. High Thoursoughput Implementation of MD5 Algorithm on GPU. In Ubiquitous Information Technologies & Applications, 2009. ICUT'09. Proceedings of the 4th International Conference on, pages 1{5. IEEE, 2010. [cited at p. 3]

32. R. Mukherjee, M. S. Rehman, K. Kothapalli, PJ Narayanan, and K. Srinathan. Presenting new Speed records and constant time encryption on the GPU. [cited at p. 3]

33. R. Zhang and X. Wang, "MD5 crack method based on compute unified device architecture," Computer Science, Vol. 38, 2011, pp. 302-305

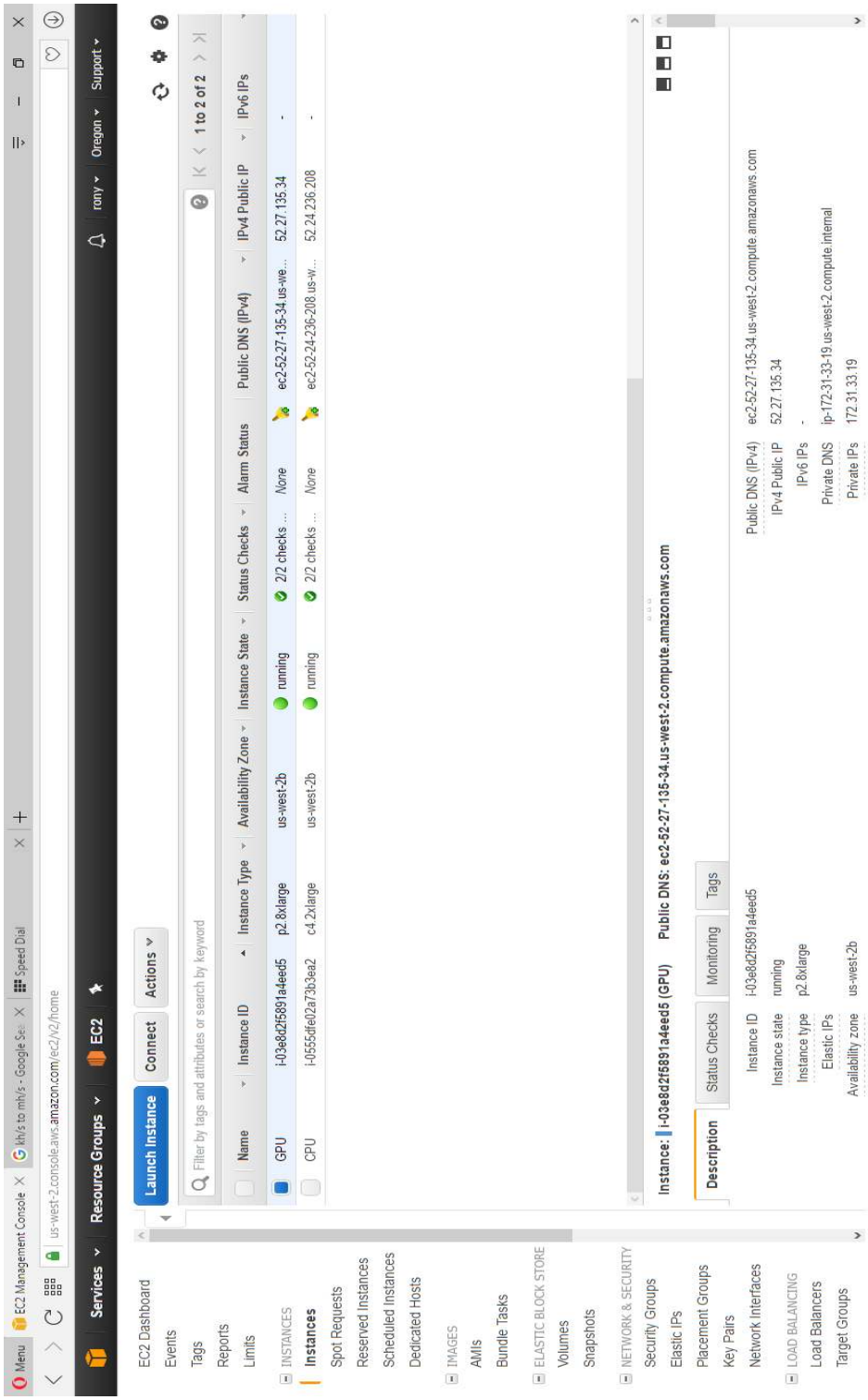
34. J. Weng, Q. Wu, and C. Yang, “OpenCL-based MD5 decryption algorithm,” Computer Engineering, Vol. 37, 2011, pp. 119-121

35. F. Wang, C. Yang, Q. Wu, and Z. Shi, “Constant memory optimizations in MD5crypt cracking algorithm on a GPU-accelerated supercomputer using CUDA,” in Proceedings of the 7th International Conference on Computer Science and Education,2012, pp. 638-642.

36. . D. H. Nguyen, T. T. Nguyen, T. N. Duong, and P. H. Pham, “Cryptanalysis of MD5 on GPU Cluster,” in Proceedings of International Conference on Information Security and Artificial Intelligence, Vol. 2, 2010, pp. 910-914

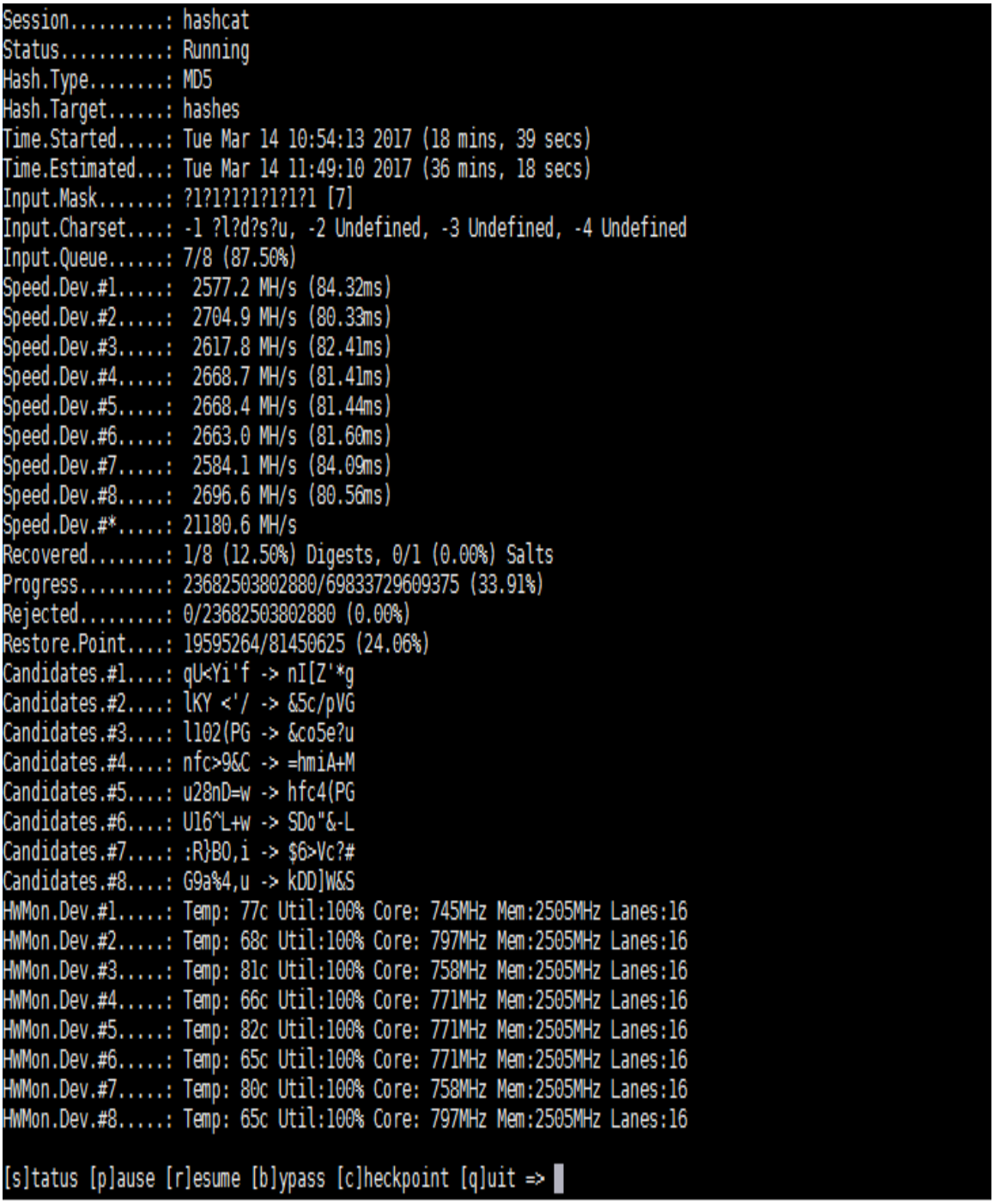
37. . R. C. Detomini, R. S. Lobato, R. Spolon, and M. A. Cavenaghi, “Using GPU to exploit parallelism on cryptography,” in Proceedings of the 6th Iberian Conference on Information Systems and Technologies, 2011, pp. 1-6

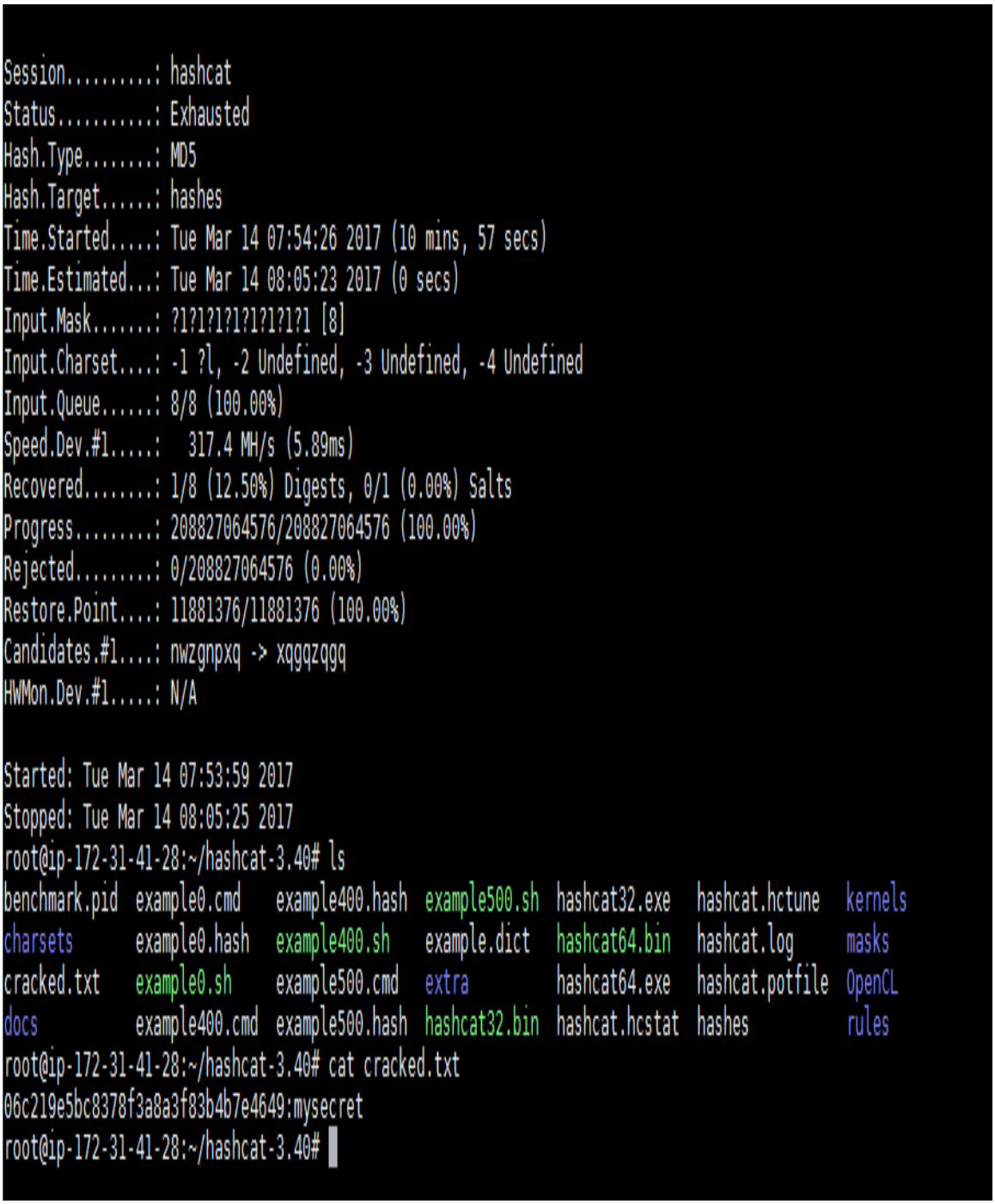
Appendix


Menu
EC2 Management Console
khi's to mhi's - Google Sea
Speed Dial
us-west-2.console.aws.amazon.com/ec2/v2/home
Services
Resource Groups
EC2
rony
Oregon
Support
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Spot Requests
Reserved Instances
Scheduled Instances
Dedicated Hosts
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
LOAD BALANCING
Load Balancers
Target Groups
Launch Instance
Connect
Actions
Filter by tags and attributes or search by keyword
1 to 2 of 2
Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS (IPv4)
IPv4 Public IP
IPv6 IPs
GPU
i-03e8d2f58891a4eed5
p2.8xlarge
us-west-2b
running
2/2 checks ...
None
ec2-52-27-135-34.us-we...
52.27.135.34
-
CPU
i-0555dfe02a73b3ea2
c4.2xlarge
us-west-2b
running
2/2 checks ...
None
ec2-52-24-236-208.us-w...
52.24.236.208
-
Instance: i-03e8d2f58891a4eed5 (GPU)
Public DNS: ec2-52-27-135-34.us-west-2.compute.amazonaws.com
Description
Status Checks
Monitoring
Tags
Instance ID
i-03e8d2f58891a4eed5
Instance state
running
Instance type
p2.8xlarge
Elastic IPs
Availability zone
us-west-2b
Public DNS (IPv4)
ec2-52-27-135-34.us-west-2.compute.amazonaws.com
IPv4 Public IP
52.27.135.34
IPv6 IPs
-
Private DNS
ip-172-31-33-19.us-west-2.compute.internal
Private IPs
172.31.33.19

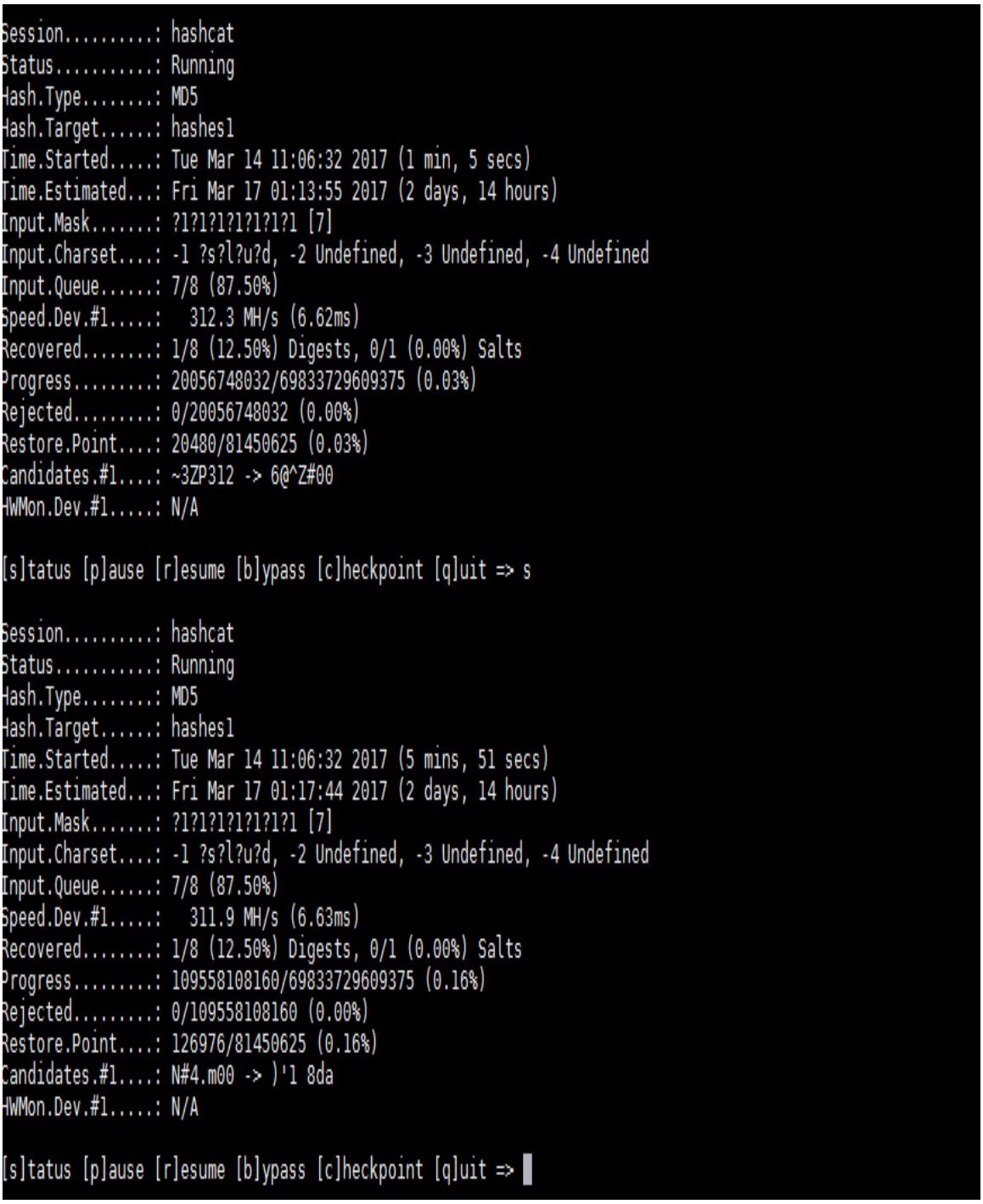
```
root@ip-172-31-33-19:~# nvidia-smi
Wed Mar 15 03:44:20 2017
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 375.26                 Driver Version: 375.26                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 0000:00:17.0     Off |                    0 |
| N/A   80C    P0   148W / 149W |    630MiB / 11439MiB |    100%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K80           Off  | 0000:00:18.0     Off |                    0 |
| N/A   62C    P0   146W / 149W |    630MiB / 11439MiB |    100%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla K80           Off  | 0000:00:19.0     Off |                    0 |
| N/A   82C    P0   146W / 149W |    630MiB / 11439MiB |    100%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla K80           Off  | 0000:00:1A.0     Off |                    0 |
| N/A   67C    P0   147W / 149W |    630MiB / 11439MiB |     99%      Default |
+-------------------------------+----------------------+----------------------+
|   4  Tesla K80           Off  | 0000:00:1B.0     Off |                    0 |
| N/A   80C    P0   142W / 149W |    630MiB / 11439MiB |    100%      Default |
+-------------------------------+----------------------+----------------------+
|   5  Tesla K80           Off  | 0000:00:1C.0     Off |                    0 |
| N/A   63C    P0   148W / 149W |    630MiB / 11439MiB |    100%      Default |
+-------------------------------+----------------------+----------------------+
|   6  Tesla K80           Off  | 0000:00:1D.0     Off |                    0 |
| N/A   82C    P0   139W / 149W |    630MiB / 11439MiB |    100%      Default |
+-------------------------------+----------------------+----------------------+
|   7  Tesla K80           Off  | 0000:00:1E.0     Off |                    0 |
| N/A   64C    P0   147W / 149W |    630MiB / 11439MiB |    100%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    0      2839    C   ./hashcat64.bin                            630MiB |
|    1      2839    C   ./hashcat64.bin                            630MiB |
|    2      2839    C   ./hashcat64.bin                            630MiB |
|    3      2839    C   ./hashcat64.bin                            630MiB |
|    4      2839    C   ./hashcat64.bin                            630MiB |
|    5      2839    C   ./hashcat64.bin                            630MiB |
|    6      2839    C   ./hashcat64.bin                            630MiB |
|    7      2839    C   ./hashcat64.bin                            630MiB |
+-----------------------------------------------------------------------------+
```

```
Session..........: hashcat
Status...........: Running
Hash.Type........: MD5
Hash.Target......: hashes
Time.Started.....: Tue Mar 14 11:50:45 2017 (12 mins, 48 secs)
Time.Estimated...: Sat Mar 18 02:51:52 2017 (3 days, 14 hours)
Input.Mask.......: ?1?1?1?1?1?1?1?1 [8]
Input.Charset....: -1 ?l?d?s?u, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue......: 8/8 (100.00%)
Speed.Dev.#1.....:  2577.2 MH/s (84.32ms)
Speed.Dev.#2.....:  2704.5 MH/s (80.34ms)
Speed.Dev.#3.....:  2616.9 MH/s (83.03ms)
Speed.Dev.#4.....:  2666.5 MH/s (81.47ms)
Speed.Dev.#5.....:  2667.9 MH/s (81.45ms)
Speed.Dev.#6.....:  2662.3 MH/s (81.63ms)
Speed.Dev.#7.....:  2584.5 MH/s (84.09ms)
Speed.Dev.#8.....:  2697.6 MH/s (80.53ms)
Speed.Dev.#*.....: 21177.5 MH/s
Recovered........: 4/8 (50.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 16257662320640/6634204312890625 (0.25%)
Rejected.........: 0/16257662320640 (0.00%)
Restore.Point....: 7667712/7737809375 (0.10%)
Candidates.#1....: ci#"-|ma -> _l$e!YCA
Candidates.#2....: \g%j.<ma -> Yn?S^+pa
Candidates.#3....: .bZ/y(ka -> zlW(X)ll
Candidates.#4....: "T/Vz~ge -> Nk=$l';;
Candidates.#5....: k3=Z2(45 -> ;-#,L<an
Candidates.#6....: T:]HB:ba -> d\;.E^ma
Candidates.#7....: o6Q]$"$$ -> s.Q`r(#l
Candidates.#8....: ?B'N,&ge -> ux;Cl^ne
HWMon.Dev.#1.....: Temp: 77c Util:100% Core: 745MHz Mem:2505MHz Lanes:16
HWMon.Dev.#2.....: Temp: 68c Util:100% Core: 797MHz Mem:2505MHz Lanes:16
HWMon.Dev.#3.....: Temp: 82c Util:100% Core: 758MHz Mem:2505MHz Lanes:16
HWMon.Dev.#4.....: Temp: 66c Util:100% Core: 771MHz Mem:2505MHz Lanes:16
HWMon.Dev.#5.....: Temp: 83c Util:100% Core: 771MHz Mem:2505MHz Lanes:16
HWMon.Dev.#6.....: Temp: 65c Util:100% Core: 771MHz Mem:2505MHz Lanes:16
HWMon.Dev.#7.....: Temp: 80c Util:100% Core: 758MHz Mem:2505MHz Lanes:16
HWMon.Dev.#8.....: Temp: 65c Util:100% Core: 797MHz Mem:2505MHz Lanes:16
```

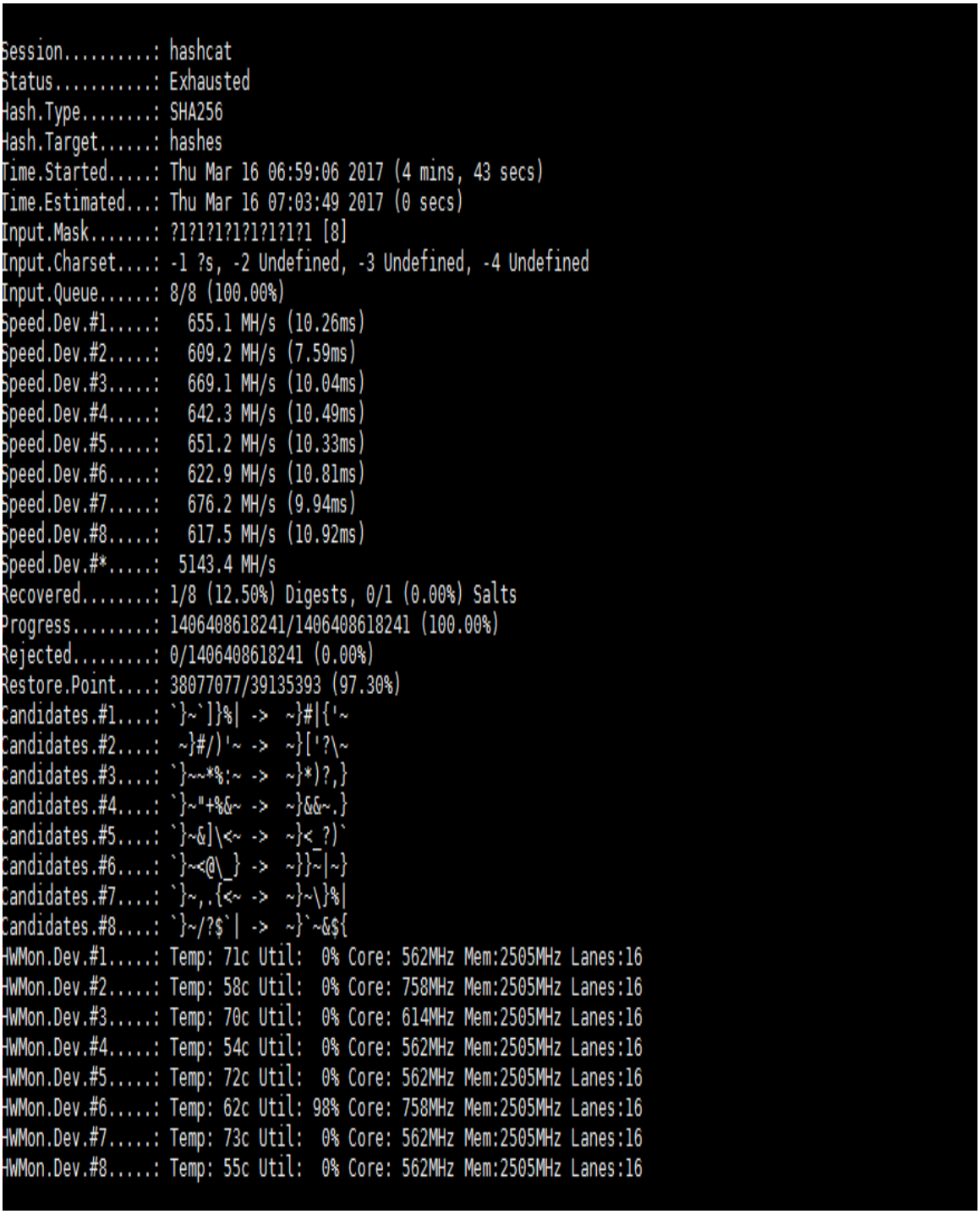

Session.....: hashcat
Status.....: Running
Hash.Type.....: MD5
Hash.Target.....: hashes
Time.Started.....: Tue Mar 14 10:54:13 2017 (18 mins, 39 secs)
Time.Estimated...: Tue Mar 14 11:49:10 2017 (36 mins, 18 secs)
Input.Mask.....: ?1?1?1?1?1?1?1 [7]
Input.Charset....: -1 ?l?d?s?u, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue.....: 7/8 (87.50%)
Speed.Dev.#1.....: 2577.2 MH/s (84.32ms)
Speed.Dev.#2.....: 2704.9 MH/s (80.33ms)
Speed.Dev.#3.....: 2617.8 MH/s (82.41ms)
Speed.Dev.#4.....: 2668.7 MH/s (81.41ms)
Speed.Dev.#5.....: 2668.4 MH/s (81.44ms)
Speed.Dev.#6.....: 2663.0 MH/s (81.60ms)
Speed.Dev.#7.....: 2584.1 MH/s (84.09ms)
Speed.Dev.#8.....: 2696.6 MH/s (80.56ms)
Speed.Dev.#*.....: 21180.6 MH/s
Recovered.....: 1/8 (12.50%) Digests, 0/1 (0.00%) Salts
Progress.....: 23682503802880/69833729609375 (33.91%)
Rejected.....: 0/23682503802880 (0.00%)
Restore.Point....: 19595264/81450625 (24.06%)
Candidates.#1....: qU<Yi'f -> nI[Z'*g
Candidates.#2....: lKY <'/ -> &5c/pVG
Candidates.#3....: l102(PG -> &co5e?u
Candidates.#4....: nfc>9&C -> =hmiA+M
Candidates.#5....: u28nD=w -> hfc4(PG
Candidates.#6....: U16^L+w -> SDo"&-L
Candidates.#7....: :R}BO,i -> $6>Vc?#
Candidates.#8....: G9a%4,u -> kDD]W&S
HWMon.Dev.#1.....: Temp: 77c Util:100% Core: 745MHz Mem:2505MHz Lanes:16
HWMon.Dev.#2.....: Temp: 68c Util:100% Core: 797MHz Mem:2505MHz Lanes:16
HWMon.Dev.#3.....: Temp: 81c Util:100% Core: 758MHz Mem:2505MHz Lanes:16
HWMon.Dev.#4.....: Temp: 66c Util:100% Core: 771MHz Mem:2505MHz Lanes:16
HWMon.Dev.#5.....: Temp: 82c Util:100% Core: 771MHz Mem:2505MHz Lanes:16
HWMon.Dev.#6.....: Temp: 65c Util:100% Core: 771MHz Mem:2505MHz Lanes:16
HWMon.Dev.#7.....: Temp: 80c Util:100% Core: 758MHz Mem:2505MHz Lanes:16
HWMon.Dev.#8.....: Temp: 65c Util:100% Core: 797MHz Mem:2505MHz Lanes:16
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>

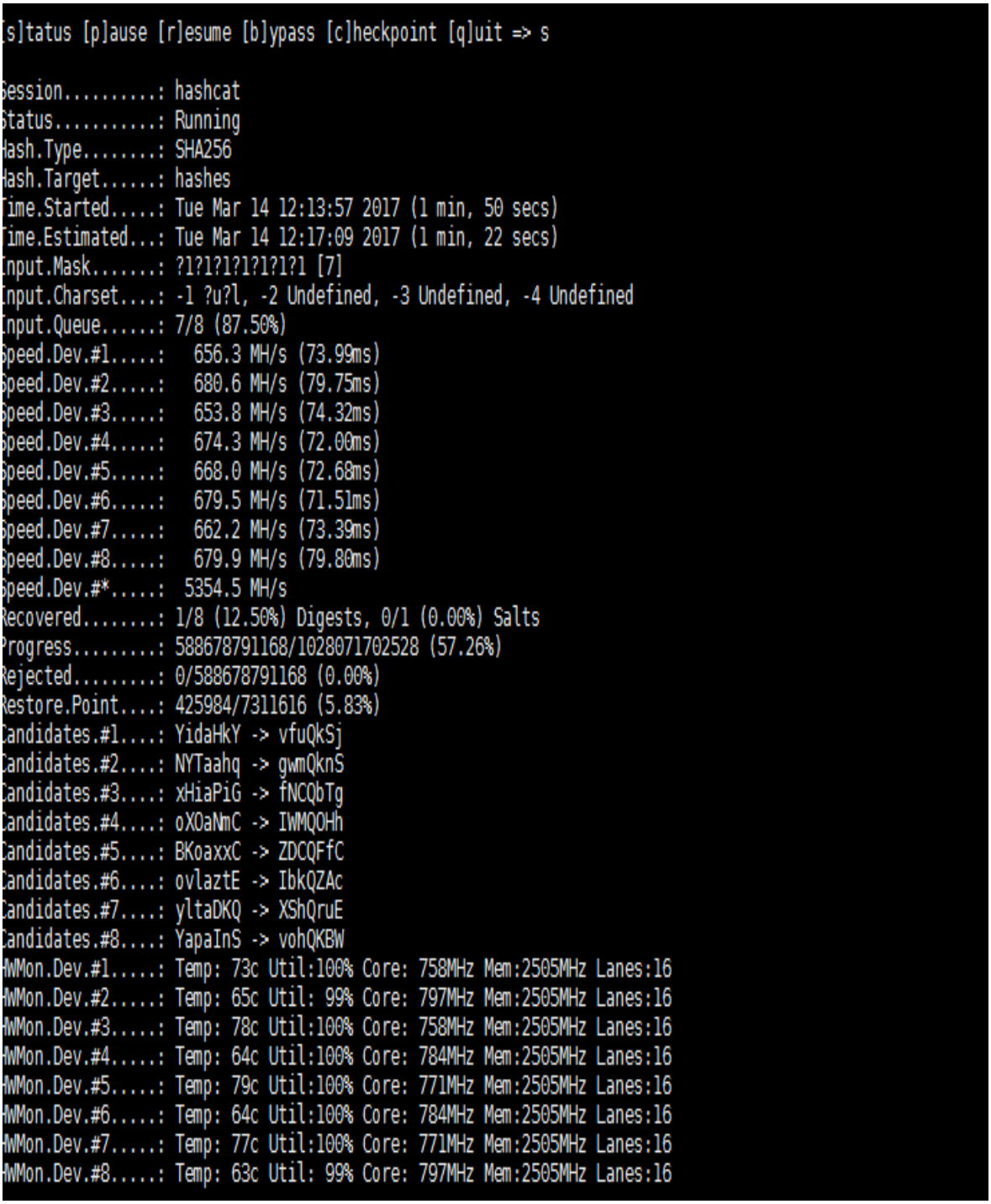

Session..........: hashcat
Status...........: Exhausted
Hash.Type........: MD5
Hash.Target......: hashes
Time.Started.....: Tue Mar 14 07:54:26 2017 (10 mins, 57 secs)
Time.Estimated...: Tue Mar 14 08:05:23 2017 (0 secs)
Input.Mask.......: ?l?l?l?l?l?l?l?l [8]
Input.Charset....: -1 ?l, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue......: 8/8 (100.00%)
Speed.Dev.#1.....:   317.4 MH/s (5.89ms)
Recovered........: 1/8 (12.50%) Digests, 0/1 (0.00%) Salts
Progress.........: 208827064576/208827064576 (100.00%)
Rejected.........: 0/208827064576 (0.00%)
Restore.Point....: 11881376/11881376 (100.00%)
Candidates.#1....: nwzgnpxq -> xqgqzqgq
HWMon.Dev.#1.....: N/A
Started: Tue Mar 14 07:53:59 2017
Stopped: Tue Mar 14 08:05:25 2017
root@ip-172-31-41-28:~/hashcat-3.40# ls
benchmark.pid example0.cmd example400.hash example500.sh hashcat32.exe hashcat.hctune kernels
charsets example0.hash example400.sh example.dict hashcat64.bin hashcat.log masks
cracked.txt example0.sh example500.cmd extra hashcat64.exe hashcat.potfile OpenCL
docs example400.cmd example500.hash hashcat32.bin hashcat.hcstat hashes rules
root@ip-172-31-41-28:~/hashcat-3.40# cat cracked.txt
06c219e5bc8378f3a8a3f83b4b7e4649:mysecret
root@ip-172-31-41-28:~/hashcat-3.40#

```
root@ip-172-31-41-28:~/hashcat-3.40# ./hashcat64.bin hashes -m 0 -a  3  ?1?1?1?1?1?1?1?1 --increment -1 ?l -o cracked.txt
hashcat (v3.40) starting...

OpenCL Platform #1: Intel(R) Corporation
========================================
* Device #1: Intel(R) Xeon(R) CPU E5-2666 v3 @ 2.90GHz, 3759/15038 MB allocatable, 8MCU

Hashes: 8 digests; 8 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates

Applicable Optimizers:
* Zero-Byte
* Precompute-Init
* Precompute-Merkle-Demgard
* Meet-In-The-Middle
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Salt
* Brute-Force
* Raw-Hash

Watchdog: Hardware Monitoring Interface not found on your system
Watchdog: Temperature abort trigger disabled
Watchdog: Temperature retain trigger disabled
```

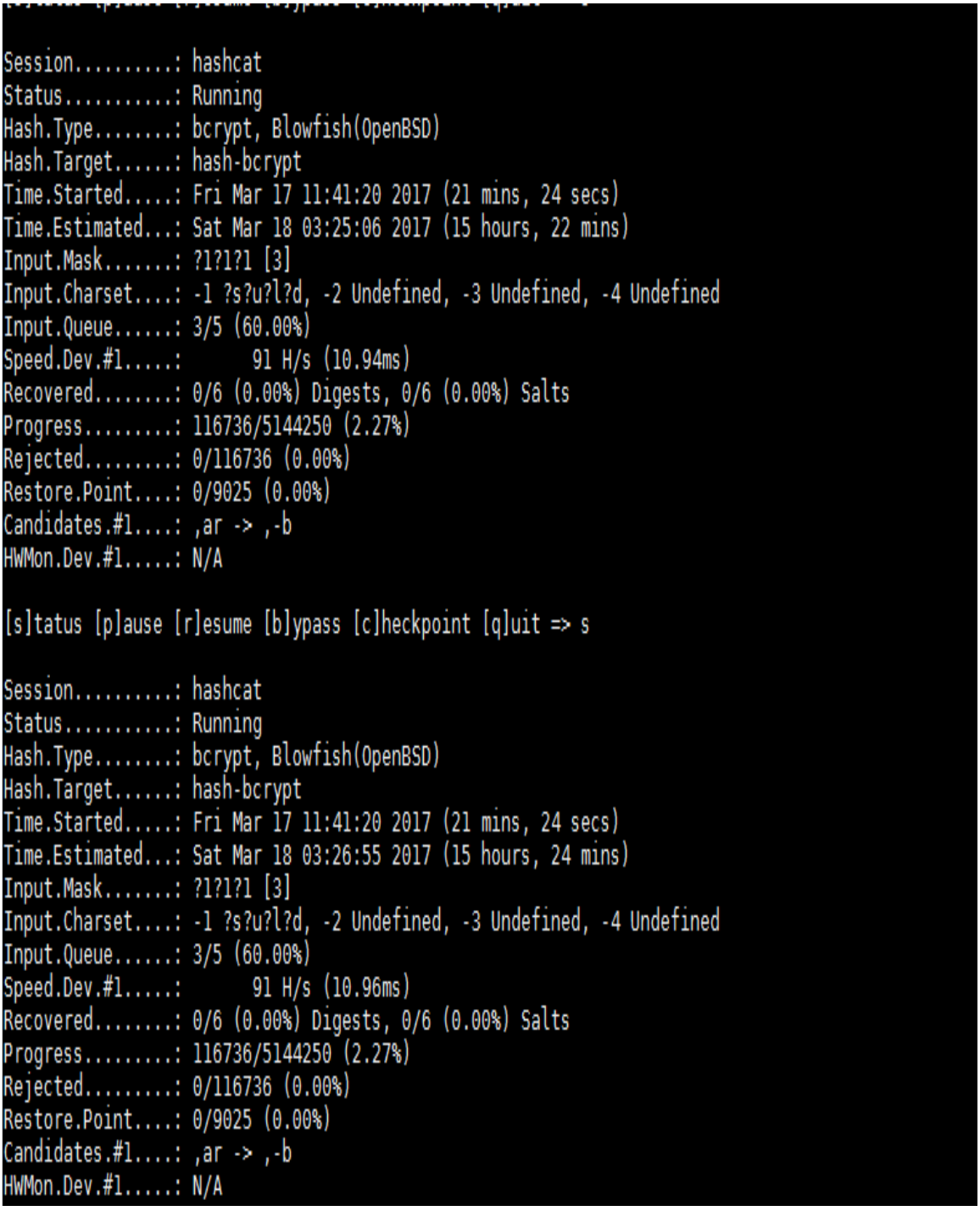

Session.....: hashcat
Status.....: Running
Hash.Type.....: MD5
Hash.Target.....: hashes1
Time.Started.....: Tue Mar 14 11:06:32 2017 (1 min, 5 secs)
Time.Estimated...: Fri Mar 17 01:13:55 2017 (2 days, 14 hours)
Input.Mask.....: ?1?1?1?1?1?1?1 [7]
Input.Charset....: -1 ?s?l?u?d, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue.....: 7/8 (87.50%)
Speed.Dev.#1.....: 312.3 MH/s (6.62ms)
Recovered.....: 1/8 (12.50%) Digests, 0/1 (0.00%) Salts
Progress.....: 20056748032/69833729609375 (0.03%)
Rejected.....: 0/20056748032 (0.00%)
Restore.Point....: 20480/81450625 (0.03%)
Candidates.#1....: ~3ZP312 -> 6@^Z#00
HWMon.Dev.#1.....: N/A
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit => s
Session.....: hashcat
Status.....: Running
Hash.Type.....: MD5
Hash.Target.....: hashes1
Time.Started.....: Tue Mar 14 11:06:32 2017 (5 mins, 51 secs)
Time.Estimated...: Fri Mar 17 01:17:44 2017 (2 days, 14 hours)
Input.Mask.....: ?1?1?1?1?1?1?1 [7]
Input.Charset....: -1 ?s?l?u?d, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue.....: 7/8 (87.50%)
Speed.Dev.#1.....: 311.9 MH/s (6.63ms)
Recovered.....: 1/8 (12.50%) Digests, 0/1 (0.00%) Salts
Progress.....: 109558108160/69833729609375 (0.16%)
Rejected.....: 0/109558108160 (0.00%)
Restore.Point....: 126976/81450625 (0.16%)
Candidates.#1....: N#4.m00 ->)'l 8da
HWMon.Dev.#1.....: N/A
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>

```
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit => s

Session..........: hashcat
Status...........: Running
Hash.Type........: SHA256
Hash.Target......: hashes
Time.Started.....: Wed Mar 15 03:03:09 2017 (2 mins, 39 secs)
Time.Estimated...: Wed Mar 15 10:47:51 2017 (7 hours, 42 mins)
Input.Mask.......: ?1?1?1?1?1?1?1 [7]
Input.Charset....: -1 ?l?s, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue......: 7/8 (87.50%)
Speed.Dev.#1.....: 89254.9 kH/s (11.65ms)
Recovered........: 0/8 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 14192809984/2488651484819 (0.57%)
Rejected.........: 0/14192809984 (0.00%)
Restore.Point....: 67584/12117361 (0.56%)
Candidates.#1....: a`:_zwa -> [}fchga
HWMon.Dev.#1.....: N/A

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit => s

Session..........: hashcat
Status...........: Running
Hash.Type........: SHA256
Hash.Target......: hashes
Time.Started.....: Wed Mar 15 03:03:09 2017 (6 mins, 23 secs)
Time.Estimated...: Wed Mar 15 10:47:57 2017 (7 hours, 38 mins)
Input.Mask.......: ?1?1?1?1?1?1?1 [7]
Input.Charset....: -1 ?l?s, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue......: 7/8 (87.50%)
Speed.Dev.#1.....: 89235.5 kH/s (11.58ms)
Recovered........: 0/8 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 34171623424/2488651484819 (1.37%)
Rejected.........: 0/34171623424 (0.00%)
Restore.Point....: 165888/12117361 (1.37%)
Candidates.#1....: =nl+"be -> xvlww<>
HWMon.Dev.#1.....: N/A

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```


Session.....: hashcat
Status.....: Exhausted
Hash.Type.....: SHA256
Hash.Target.....: hashes
Time.Started.....: Thu Mar 16 06:59:06 2017 (4 mins, 43 secs)
Time.Estimated...: Thu Mar 16 07:03:49 2017 (0 secs)
Input.Mask.....: ?1?1?1?1?1?1?1?1 [8]
Input.Charset....: -1 ?s, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue.....: 8/8 (100.00%)
Speed.Dev.#1.....: 655.1 MH/s (10.26ms)
Speed.Dev.#2.....: 609.2 MH/s (7.59ms)
Speed.Dev.#3.....: 669.1 MH/s (10.04ms)
Speed.Dev.#4.....: 642.3 MH/s (10.49ms)
Speed.Dev.#5.....: 651.2 MH/s (10.33ms)
Speed.Dev.#6.....: 622.9 MH/s (10.81ms)
Speed.Dev.#7.....: 676.2 MH/s (9.94ms)
Speed.Dev.#8.....: 617.5 MH/s (10.92ms)
Speed.Dev.#*.....: 5143.4 MH/s
Recovered.....: 1/8 (12.50%) Digests, 0/1 (0.00%) Salts
Progress.....: 1406408618241/1406408618241 (100.00%)
Rejected.....: 0/1406408618241 (0.00%)
Restore.Point....: 38077077/39135393 (97.30%)
Candidates.#1....: `}~`]}%| -> ~}#|{'~
Candidates.#2....: ~}#/)'~ -> ~}['?\~
Candidates.#3....: `}~~*%:~ -> ~}*)?,}
Candidates.#4....: `}~"+%&~ -> ~}&&~.}
Candidates.#5....: `}~&]\<~ -> ~}<_?)`
Candidates.#6....: `}~<@_} -> ~}}~|~}
Candidates.#7....: `}~,.{<~ -> ~}~\}%|
Candidates.#8....: `}~/?$`| -> ~}`~&${
HWMon.Dev.#1.....: Temp: 71c Util: 0% Core: 562MHz Mem:2505MHz Lanes:16
HWMon.Dev.#2.....: Temp: 58c Util: 0% Core: 758MHz Mem:2505MHz Lanes:16
HWMon.Dev.#3.....: Temp: 70c Util: 0% Core: 614MHz Mem:2505MHz Lanes:16
HWMon.Dev.#4.....: Temp: 54c Util: 0% Core: 562MHz Mem:2505MHz Lanes:16
HWMon.Dev.#5.....: Temp: 72c Util: 0% Core: 562MHz Mem:2505MHz Lanes:16
HWMon.Dev.#6.....: Temp: 62c Util: 98% Core: 758MHz Mem:2505MHz Lanes:16
HWMon.Dev.#7.....: Temp: 73c Util: 0% Core: 562MHz Mem:2505MHz Lanes:16
HWMon.Dev.#8.....: Temp: 55c Util: 0% Core: 562MHz Mem:2505MHz Lanes:16

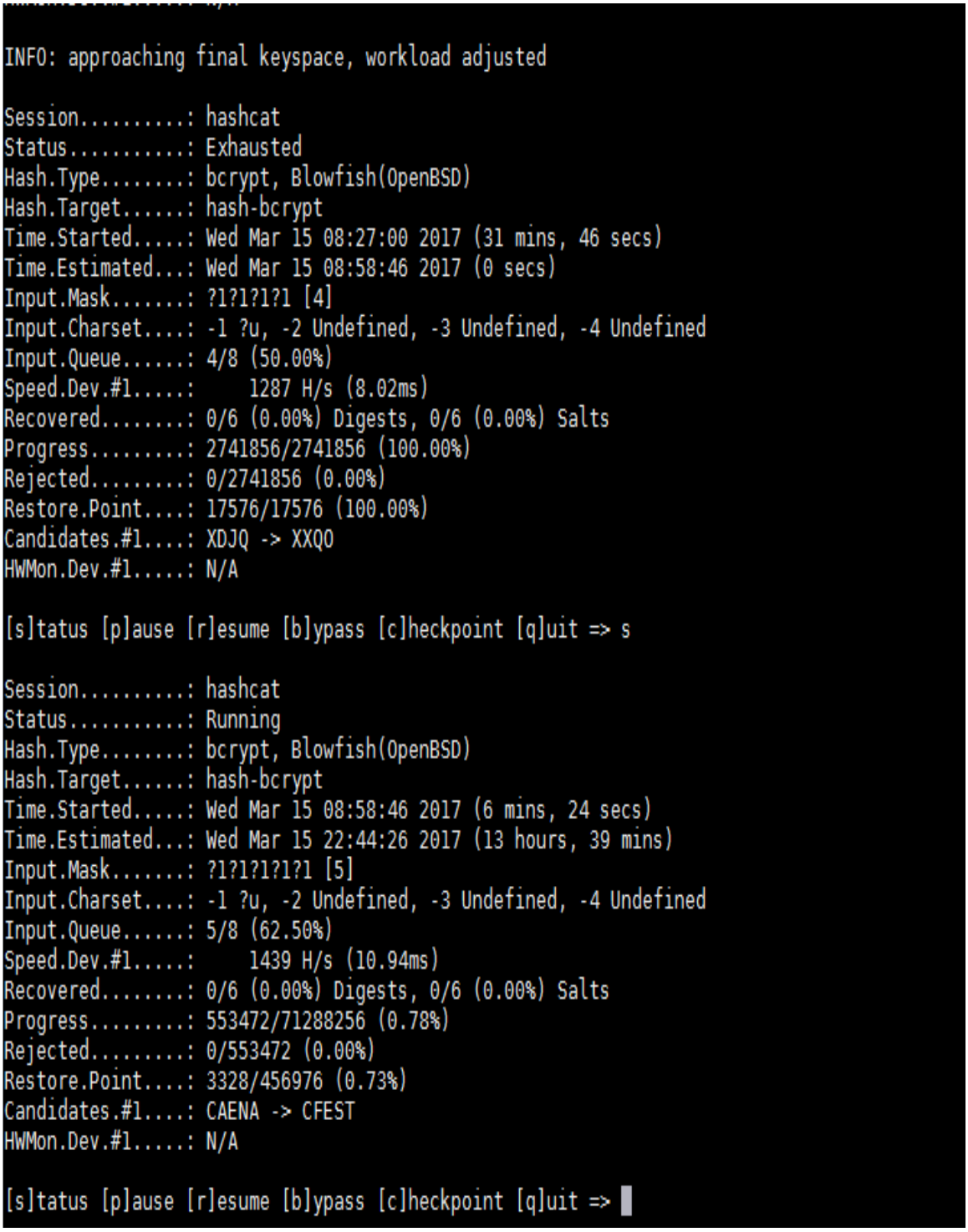

s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit => s
Session.....: hashcat
Status.....: Running
Hash.Type.....: SHA256
Hash.Target.....: hashes
Time.Started.....: Tue Mar 14 12:13:57 2017 (1 min, 50 secs)
Time.Estimated...: Tue Mar 14 12:17:09 2017 (1 min, 22 secs)
Input.Mask.....: ?1?1?1?1?1?1?1?1 [7]
Input.Charset....: -1 ?u?l, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue.....: 7/8 (87.50%)
Speed.Dev.#1.....: 656.3 MH/s (73.99ms)
Speed.Dev.#2.....: 680.6 MH/s (79.75ms)
Speed.Dev.#3.....: 653.8 MH/s (74.32ms)
Speed.Dev.#4.....: 674.3 MH/s (72.00ms)
Speed.Dev.#5.....: 668.0 MH/s (72.68ms)
Speed.Dev.#6.....: 679.5 MH/s (71.51ms)
Speed.Dev.#7.....: 662.2 MH/s (73.39ms)
Speed.Dev.#8.....: 679.9 MH/s (79.80ms)
Speed.Dev.#*.....: 5354.5 MH/s
Recovered.....: 1/8 (12.50%) Digests, 0/1 (0.00%) Salts
Progress.....: 588678791168/1028071702528 (57.26%)
Rejected.....: 0/588678791168 (0.00%)
Restore.Point....: 425984/7311616 (5.83%)
Candidates.#1....: YidaHkY -> vfuQkSj
Candidates.#2....: NYTaahq -> gwmQknS
Candidates.#3....: xHiaPiG -> fNCQbTg
Candidates.#4....: oXOaNmC -> IWMQOHh
Candidates.#5....: BKoaxxC -> ZDCQFfC
Candidates.#6....: ovlaztE -> IbkQZAc
Candidates.#7....: yltaDKQ -> XShQruE
Candidates.#8....: YapaInS -> vohQKBW
HWMon.Dev.#1.....: Temp: 73c Util:100% Core: 758MHz Mem:2505MHz Lanes:16
HWMon.Dev.#2.....: Temp: 65c Util: 99% Core: 797MHz Mem:2505MHz Lanes:16
HWMon.Dev.#3.....: Temp: 78c Util:100% Core: 758MHz Mem:2505MHz Lanes:16
HWMon.Dev.#4.....: Temp: 64c Util:100% Core: 784MHz Mem:2505MHz Lanes:16
HWMon.Dev.#5.....: Temp: 79c Util:100% Core: 771MHz Mem:2505MHz Lanes:16
HWMon.Dev.#6.....: Temp: 64c Util:100% Core: 784MHz Mem:2505MHz Lanes:16
HWMon.Dev.#7.....: Temp: 77c Util:100% Core: 771MHz Mem:2505MHz Lanes:16
HWMon.Dev.#8.....: Temp: 63c Util: 99% Core: 797MHz Mem:2505MHz Lanes:16

```
Session..........: hashcat
Status...........: Running
Hash.Type........: SHA256
Hash.Target......: hashes
Time.Started.....: Wed Mar 15 03:10:27 2017 (57 secs)
Time.Estimated...: Wed Mar 15 03:18:27 2017 (7 mins, 3 secs)
Input.Mask.......: ?1?1?1?1?1?1?1?1 [7]
Input.Charset....: -1 ?s?l, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue......: 7/8 (87.50%)
Speed.Dev.#1.....:   639.9 MH/s (84.80ms)
Speed.Dev.#2.....:   643.6 MH/s (84.36ms)
Speed.Dev.#3.....:   658.7 MH/s (82.39ms)
Speed.Dev.#4.....:   631.3 MH/s (85.85ms)
Speed.Dev.#5.....:   665.4 MH/s (81.55ms)
Speed.Dev.#6.....:   638.1 MH/s (84.94ms)
Speed.Dev.#7.....:   657.0 MH/s (82.58ms)
Speed.Dev.#8.....:   632.9 MH/s (85.78ms)
Speed.Dev.#*.....:  5166.9 MH/s
Recovered........: 0/8 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 299729158144/2488651484819 (12.04%)
Rejected.........: 0/299729158144 (0.00%)
Restore.Point....: 0/12117361 (0.00%)
Candidates.#1....: egqanan -> xbqnwhi
Candidates.#2....: p{"f"cc -> naqjlxu
Candidates.#3....: /$!-)gm -> ^"u ~#o
Candidates.#4....: h#!p-gl -> y$wv`\t
Candidates.#5....: ,i@xbqn -> 'n@.+_j
Candidates.#6....: q\'kuky -> _=xu$bh
Candidates.#7....: su$tkol -> drxs(^%
Candidates.#8....: u,wo;th -> (qvcrly
HWMon.Dev.#1.....: Temp: 81c Util:100% Core: 745MHz Mem:2505MHz Lanes:16
HWMon.Dev.#2.....: Temp: 67c Util: 99% Core: 745MHz Mem:2505MHz Lanes:16
HWMon.Dev.#3.....: Temp: 84c Util:100% Core: 758MHz Mem:2505MHz Lanes:16
HWMon.Dev.#4.....: Temp: 70c Util: 99% Core: 745MHz Mem:2505MHz Lanes:16
HWMon.Dev.#5.....: Temp: 81c Util: 99% Core: 771MHz Mem:2505MHz Lanes:16
HWMon.Dev.#6.....: Temp: 67c Util:100% Core: 732MHz Mem:2505MHz Lanes:16
HWMon.Dev.#7.....: Temp: 82c Util:100% Core: 771MHz Mem:2505MHz Lanes:16
HWMon.Dev.#8.....: Temp: 67c Util:100% Core: 732MHz Mem:2505MHz Lanes:16
```


Session.....: hashcat
Status.....: Running
Hash.Type.....: bcrypt, Blowfish(OpenBSD)
Hash.Target.....: hash-bcrypt
Time.Started.....: Fri Mar 17 11:41:20 2017 (21 mins, 24 secs)
Time.Estimated...: Sat Mar 18 03:25:06 2017 (15 hours, 22 mins)
Input.Mask.....: ?l?l?l [3]
Input.Charset....: -1 ?s?u?l?d, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue.....: 3/5 (60.00%)
Speed.Dev.#1.....: 91 H/s (10.94ms)
Recovered.....: 0/6 (0.00%) Digests, 0/6 (0.00%) Salts
Progress.....: 116736/5144250 (2.27%)
Rejected.....: 0/116736 (0.00%)
Restore.Point....: 0/9025 (0.00%)
Candidates.#1....: ,ar -> ,-b
HWMon.Dev.#1.....: N/A
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit => s
Session.....: hashcat
Status.....: Running
Hash.Type.....: bcrypt, Blowfish(OpenBSD)
Hash.Target.....: hash-bcrypt
Time.Started.....: Fri Mar 17 11:41:20 2017 (21 mins, 24 secs)
Time.Estimated...: Sat Mar 18 03:26:55 2017 (15 hours, 24 mins)
Input.Mask.....: ?l?l?l [3]
Input.Charset....: -1 ?s?u?l?d, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue.....: 3/5 (60.00%)
Speed.Dev.#1.....: 91 H/s (10.96ms)
Recovered.....: 0/6 (0.00%) Digests, 0/6 (0.00%) Salts
Progress.....: 116736/5144250 (2.27%)
Rejected.....: 0/116736 (0.00%)
Restore.Point....: 0/9025 (0.00%)
Candidates.#1....: ,ar -> ,-b
HWMon.Dev.#1.....: N/A

```
Session..........: hashcat
Status...........: Running
Hash.Type........: bcrypt, Blowfish(OpenBSD)
Hash.Target......: hash-bcrypt
Time.Started.....: Thu Mar 16 12:39:49 2017 (5 mins, 31 secs)
Time.Estimated...: Thu Mar 16 15:17:05 2017 (2 hours, 31 mins)
Input.Mask.......: ?1?1?1 [3]
Input.Charset....: -1 ?l?u?s?d, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue......: 3/5 (60.00%)
Speed.Dev.#1.....:       86 H/s (37.24ms)
Speed.Dev.#2.....:       66 H/s (49.16ms)
Speed.Dev.#3.....:       66 H/s (49.15ms)
Speed.Dev.#4.....:       66 H/s (48.98ms)
Speed.Dev.#5.....:       64 H/s (25.12ms)
Speed.Dev.#6.....:       66 H/s (48.89ms)
Speed.Dev.#7.....:       66 H/s (48.89ms)
Speed.Dev.#8.....:       66 H/s (48.87ms)
Speed.Dev.#*.....:      545 H/s
Recovered........: 0/6 (0.00%) Digests, 0/6 (0.00%) Salts
Progress.........: 179296/5144250 (3.49%)
Rejected.........: 0/179296 (0.00%)
Restore.Point....: 0/9025 (0.00%)
Candidates.#1....: Bar -> B r
Candidates.#2....: Wi6 -> WIg
Candidates.#3....: W!S -> We9
Candidates.#4....: Wd3 -> WQK
Candidates.#5....: E:h -> Eb6
Candidates.#6....: W8c -> WBO
Candidates.#7....: WCI -> W['
Candidates.#8....: W/5 -> W3a
HWMon.Dev.#1.....: Temp: 79c Util: 99% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#2.....: Temp: 58c Util: 99% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#3.....: Temp: 73c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#4.....: Temp: 56c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#5.....: Temp: 77c Util: 99% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#6.....: Temp: 58c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#7.....: Temp: 81c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#8.....: Temp: 59c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
```

```
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit => s

Session..........: hashcat
Status...........: Running
Hash.Type........: bcrypt, Blowfish(OpenBSD)
Hash.Target......: hash-bcrypt
Time.Started.....: Fri Mar 17 10:33:22 2017 (11 secs)
Time.Estimated...: Wed Mar 29 17:41:10 2017 (12 days, 7 hours)
Input.Mask.......: ?1?1?1?1 [4]
Input.Charset....: -1 ?u?l?s?d, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue......: 4/6 (66.67%)
Speed.Dev.#1.....:       58 H/s (49.01ms)
Speed.Dev.#2.....:       58 H/s (49.32ms)
Speed.Dev.#3.....:       57 H/s (48.71ms)
Speed.Dev.#4.....:       57 H/s (48.83ms)
Speed.Dev.#5.....:       58 H/s (48.99ms)
Speed.Dev.#6.....:       57 H/s (48.90ms)
Speed.Dev.#7.....:       58 H/s (49.10ms)
Speed.Dev.#8.....:       57 H/s (48.91ms)
Speed.Dev.#*.....:      460 H/s
Recovered........: 0/6 (0.00%) Digests, 0/6 (0.00%) Salts
Progress.........: 3328/488703750 (0.00%)
Rejected.........: 0/3328 (0.00%)
Restore.Point....: 0/857375 (0.00%)
Candidates.#1....: mari -> mxtt
Candidates.#2....: mOOR -> m ri
Candidates.#3....: m!SS -> me91
Candidates.#4....: m:ha -> mb66
Candidates.#5....: md34 -> mQKE
Candidates.#6....: mCIS -> m['s
Candidates.#7....: m/55 -> m3an
Candidates.#8....: m8ck -> mBOR
HWMon.Dev.#1.....: Temp: 64c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#2.....: Temp: 53c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#3.....: Temp: 77c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#4.....: Temp: 61c Util: 99% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#5.....: Temp: 76c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#6.....: Temp: 56c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#7.....: Temp: 77c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#8.....: Temp: 58c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
```


INFO: approaching final keyspace, workload adjusted
Session.....: hashcat
Status.....: Exhausted
Hash.Type.....: bcrypt, Blowfish(OpenBSD)
Hash.Target.....: hash-bcrypt
Time.Started.....: Wed Mar 15 08:27:00 2017 (31 mins, 46 secs)
Time.Estimated...: Wed Mar 15 08:58:46 2017 (0 secs)
Input.Mask.....: ?1?1?1?1 [4]
Input.Charset....: -1 ?u, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue.....: 4/8 (50.00%)
Speed.Dev.#1.....: 1287 H/s (8.02ms)
Recovered.....: 0/6 (0.00%) Digests, 0/6 (0.00%) Salts
Progress.....: 2741856/2741856 (100.00%)
Rejected.....: 0/2741856 (0.00%)
Restore.Point....: 17576/17576 (100.00%)
Candidates.#1....: XDJQ -> XXQO
HWMon.Dev.#1.....: N/A
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit => s
Session.....: hashcat
Status.....: Running
Hash.Type.....: bcrypt, Blowfish(OpenBSD)
Hash.Target.....: hash-bcrypt
Time.Started.....: Wed Mar 15 08:58:46 2017 (6 mins, 24 secs)
Time.Estimated...: Wed Mar 15 22:44:26 2017 (13 hours, 39 mins)
Input.Mask.....: ?1?1?1?1?1 [5]
Input.Charset....: -1 ?u, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue.....: 5/8 (62.50%)
Speed.Dev.#1.....: 1439 H/s (10.94ms)
Recovered.....: 0/6 (0.00%) Digests, 0/6 (0.00%) Salts
Progress.....: 553472/71288256 (0.78%)
Rejected.....: 0/553472 (0.00%)
Restore.Point....: 3328/456976 (0.73%)
Candidates.#1....: CAENA -> CFEST
HWMon.Dev.#1.....: N/A
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>

```
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit => s

Session..........: hashcat
Status...........: Running
Hash.Type........: bcrypt, Blowfish(OpenBSD)
Hash.Target......: hash-bcrypt
Time.Started.....: Wed Mar 15 10:40:07 2017 (18 mins, 7 secs)
Time.Estimated...: Sat Mar 18 14:49:48 2017 (3 days, 3 hours)
Input.Mask.......: ?1?1?1?1?1 [5]
Input.Charset....: -1 ?u?l, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue......: 5/8 (62.50%)
Speed.Dev.#1.....:     1043 H/s (48.89ms)
Speed.Dev.#2.....:     1045 H/s (48.89ms)
Speed.Dev.#3.....:     1031 H/s (49.46ms)
Speed.Dev.#4.....:     1040 H/s (49.06ms)
Speed.Dev.#5.....:     1044 H/s (48.90ms)
Speed.Dev.#6.....:     1041 H/s (49.01ms)
Speed.Dev.#7.....:     1043 H/s (48.89ms)
Speed.Dev.#8.....:     1032 H/s (49.47ms)
Speed.Dev.#*.....:     8320 H/s
Recovered........: 0/6 (0.00%) Digests, 0/6 (0.00%) Salts
Progress.........: 9051328/2281224192 (0.40%)
Rejected.........: 0/9051328 (0.00%)
Restore.Point....: 23712/7311616 (0.32%)
Candidates.#1....: Cadge -> CXDST
Candidates.#2....: PaYTE -> PXvma
Candidates.#3....: VaEDE -> VXske
Candidates.#4....: vaope -> vXUDE
Candidates.#5....: Cahst -> CXKHA
Candidates.#6....: vaBri -> vXxus
Candidates.#7....: Carst -> CXAIN
Candidates.#8....: OaJMA -> OXlya
HWMon.Dev.#1.....: Temp: 67c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#2.....: Temp: 51c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#3.....: Temp: 72c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#4.....: Temp: 54c Util: 99% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#5.....: Temp: 69c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#6.....: Temp: 53c Util: 99% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#7.....: Temp: 69c Util:100% Core: 875MHz Mem:2505MHz Lanes:16
HWMon.Dev.#8.....: Temp: 56c Util:100% Core: 875MHz Mem:2505MHz Lanes:16

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

```
Input.Charset....: -1 ?u?l, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue......: 3/5 (60.00%)
Speed.Dev.#1.....:       71 H/s (7.82ms)
Recovered........: 0/6 (0.00%) Digests, 0/6 (0.00%) Salts
Progress.........: 822768/843648 (97.53%)
Rejected.........: 0/822768 (0.00%)
Restore.Point....: 2560/2704 (94.67%)
Candidates.#1....: gcX -> gXv
HWMon.Dev.#1.....: N/A

Session..........: hashcat
Status...........: Exhausted
Hash.Type........: bcrypt, Blowfish(OpenBSD)
Hash.Target......: hash-bcrypt
Time.Started.....: Fri Mar 17 12:08:04 2017 (2 hours, 36 mins)
Time.Estimated...: Fri Mar 17 14:44:51 2017 (0 secs)
Input.Mask.......: ?1?1?1 [3]
Input.Charset....: -1 ?u?l, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue......: 3/5 (60.00%)
Speed.Dev.#1.....:       72 H/s (7.81ms)
Recovered........: 0/6 (0.00%) Digests, 0/6 (0.00%) Salts
Progress.........: 843648/843648 (100.00%)
Rejected.........: 0/843648 (0.00%)
Restore.Point....: 2704/2704 (100.00%)
Candidates.#1....: XcX -> XXv
HWMon.Dev.#1.....: N/A

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit => s

Session..........: hashcat
Status...........: Running
Hash.Type........: bcrypt, Blowfish(OpenBSD)
Hash.Target......: hash-bcrypt
Time.Started.....: Fri Mar 17 14:44:54 2017 (17 mins, 37 secs)
Time.Estimated...: Tue Mar 21 08:20:07 2017 (3 days, 17 hours)
Input.Mask.......: ?1?1?1?1 [4]
Input.Charset....: -1 ?u?l, -2 Undefined, -3 Undefined, -4 Undefined
Input.Queue......: 4/5 (80.00%)
Speed.Dev.#1.....:       91 H/s (10.94ms)
Recovered........: 2/6 (33.33%) Digests, 2/6 (33.33%) Salts
Progress.........: 127488/43869696 (0.29%)
Rejected.........: 0/127488 (0.00%)
Restore.Point....: 256/140608 (0.18%)
Candidates.#1....: JVan -> JJJA
HWMon.Dev.#1.....: N/A
```