University Of Piraeus

Department Of Digital Systems

Postgraduate Programme " Techno-Economic Management and Security Of Digital Systems "

MASTER THESIS

SUBJECT: PASSWORD CRACKING

Supervisor:	**Xenakis Christos**
Student:	Christofakis Michail
A.M:	MTE1138

PIRAEUS

JULY 2013

Acknowledgements

I would like to thank my teacher and supervisor for this paper, Mr. Christos Xenakis, as for his help in the accomplishment of this dissertation. Also I would like to thank my family for the whole support all this years in giving me the strength to continue making my dreams come true.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Abstract

Information security is the next big thing in computers society because of the rapidly growing security incidents and the outcomes of those. Hacking and cracking existed even from the start of the eighties decade when there was the first step of the interconnection through the internet between humans. From then and ever after there was a big explosion of such incidents mostly because of the worldwide web which was introduced in the early nineties. Following the huge steps forward of computers evolution till today anyone can now clearly see that security is more than ever important to everyday life. Mainly because everything happens really fast and everything is done online, from communicating with each other to transactions and so on. Finally the possibilities plus the software and hardware offered are endless today, something that gives the opportunity to a malicious person to do some significant damage. Last but not least, attention should be given to security measures and techniques to avoid unhappy situations and security awareness should be a part of everyone's life where no one is excluded.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Summary

This paper refers to the use of passwords on computer systems and the ways of attacking them. It also states ways of protecting passwords from cracking and puts through serious testing on some cracking tools. Afterwards it focuses on comparing the efficiency of these cracking tools and it draws some conclusions on their use. Finally future work is left to be done in order to make even greater steps through password security.

In the first chapter, a general reference to passwords is made as to what they are and what they consist of. Also an overview is made on the importance of having secure passwords nowadays. Lastly the main methods of attacking passwords are presented.

In the second chapter, some ways of making password cracking harder are presented and emphasis is given on how to secure and strengthen passwords in general using salts, key stretching and considering password entropy. Also the classes of attack are presented and the time needed to crack passwords is explained. Moreover, password policies are introduced and guidelines for having secure passwords are given.

In the third chapter, some ways to accelerate password cracking are shown and an extensive reference on password cracking tools is made. Also a thorough overview on basic usage is given for four basic and commonly used tools.

In the fourth chapter, a hands on experience is demonstrated on how to crack hashes using basic password cracking tools with an emphasis on hashcat tool. It is taken from 2012 contest of Korelogic named CrackMeIfYouCan. There was given several types of hashes and it was requested to crack as many as possible in two days time. So a complete task is presented here on specific hash types like md5.

In the final chapter of this paper, the results drawn from the above procedures are gathered and put together in order to be analyzed. Several comparison graphs are made that give valuable information to the reader as to what these tools can do. Lastly special conclusions and suggestions are given considering the whole aspect of password cracking thing and future work is anticipated in the hope of perfectioning the knowledge on the field.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Table Of Contents

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Figure Table

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Introduction

On February 2004, Bill Gates gave a speech in the RSA Security Conference predicting the death of passwords. In the long term, everything he said is mostly true ,but when comparing to nowadays ,passwords still seems to be very much alive. They are the most used means of authentication even though there have been throughout the years a lot of efforts to bypass them.

Many huge companies have demonstrated alternative ways of authentication but very few are in full use today. It seems that people have not overcome passwords and that companies still can't find an efficient workaround. Despite a growing number of graphical and biometric authentication mechanisms, passwords remain the most familiar and commonly-used form of user authentication in organizational settings. This is why security breaches happen more often and the outcome is many times devastating. Information technology is moving forward so quickly that password techniques and policies are looking obsolete. This is where password cracking emerges and hackers take full advantage of the weak protection of critical information.

In cryptanalysis and security of information systems, Password Cracking is the process of recovering passwords from data that has been stored or transmitted by a computer system.(1) A common approach is for the attacker to try different possible guesses to guess the password of a user. Another common approach is to assume that the user has 'forgotten' his password and calls for a change from a service (eg to send a new password to the email account of his own).

The purpose of Password Cracking could help a user recover a forgotten password (though creating a new password is less of a security risk, but requires the intervention of an administrator) to gain unauthorized access to a computer system or as a preventive measure by the administrators of a computer system in order to search for easy passwords.

In these days of ultimate information gathered from every possible source, people find it hard to remember all passwords that are needed in their everyday life. They have a lot of user accounts in different services across the web and many more on their mobile devices. As information technology moves forward it will be harder and harder to remember passwords because of the increase in services and accounts.

That said it is very pleasing for malicious users of the web to emerge and try to crack these passwords in every imaginable way. There are many open source tools and services for that reason and if misused they can cause real damage to every aspect of the victims life. Many times harm can be done not only to individuals but to organizations and businesses in general. And there is where the effects of these actions count the most. We have seen through history a great amount of cases that password cracking did matter a lot.

In the next chapters of this paper, a quality overview is being given to passwords in general. An effort is made to present all aspects of password security. Password cracking techniques are presented also in order to help the reader understand how easy it can be to 'crack a password'. Finally, a research have taken place as to testing the efficiency and speed of well known password cracking tools when having different types of hashes, and there have been some suggestions ,plus more room for further research is given.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

1.INTRODUCTION TO PASSWORDS

1.1 What is a password

A password is a word or sequence of characters entered, often with a user name, to a computer system so that it can be connected to the user or to access a resource. Passwords are a common form of validation (authentication). For complete safety the password has to be kept secret from users who have not been given access.

The use of passwords has existed since ancient times. The guards of a place knew a password or a slogan that was given to them ..They would allow access only to persons who knew the code. In modern times, passwords are used to control access to protected computer operating systems, on mobile phones, cable TV decoders, in automated machines (ATMs), etc. A typical computer user may require passwords for many purposes: to access accounts of operating systems, for retrieving email from servers, to have access to programs, databases, networks, on websites, even reading the newspaper on-line.

Despite the name, there is no need for passwords to be actual words. Passwords that are not real words are more difficult to find by malicious users. Some passwords consist of several words and are called passphrases. The term Passcode is used when the secret information is protected by numbers, such as a personal identification number (PIN) commonly used for accessing ATM's. Passwords are generally short in order to be able to be memorized.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

1.2 Passphrase & password

A passphrase is a sequence of words or other text used to control access to a computer system, program or file mostly in cryptographic programs and systems. The passphrase is similar to a password in using it, but it is generally longer and harder for extra security. It is an extended version of a password ,typically consisting of multiple words. Because of this, it is more secure against "dictionary attacks". It is relatively long and contains a combination of uppercase and lowercase letters and numeric and symbolic characters.

The Passphrases are particularly used in systems which use them as encryption key. They are generally used for the certification of public / private key. A public / private key system defines the mathematical relationship between the public key known to everyone and a private key that is known only to one user. Without Passphrases to unlock the private key, the user cannot gain access.

1.3 Introduction to password cracking

The security of a system which is password protected depends on several factors. The system should be designed for sound security, with protection against computer malware and malicious users. Here are some specific password management issues that must be considered, such as the extent to which an attacker can guess passwords.

The extent to which an attacker can guess passwords of a system is a key factor in securing systems. Some systems require a time of several seconds after a small number (eg, three) of failed password entry attempts. Under other circumstances, however, such systems can be quite secure with relatively simple passwords, if they have been chosen well and can not be easily guessed.

Many systems store or transmit cryptographic hash of the password in a way that makes the hash value accessible to an attacker. When this is done, an attacker can work offline, reviewing prospective codes in order to find the real password. The codes used to generate cryptographic keys (eg disk encryption or security of WI-FI) can also be guessed easily. The lists of common passwords are widely available and can make password attacks very efficient. Security in such situations depends on the complexity of the passwords or passphrases which makes such an attack computationally infeasible for the attacker. Some systems, such as PGP and Wi Fi WPA implement computation-intensive hash to the password to slow down such attacks.

Password cracking then is essentially the process of recovering passwords from files / data stored in a computer system. A common approach is to constantly guess passwords for identification purposes and finding the desired password. The purpose may be to recover a forgotten password from a user to access a system. It can be used both by people who have good will in order to help find information for example in investigations ,or by malicious people who intend to gain unauthorized access and cause damage to computer systems.

1.4 Main Methods of Attack

There are several methods of attacking a computer system in order to steal passwords or generally treat it in a bad manner. The main and most used methods are:

1.4.1 Weak encryption

If a system uses a reversible function to conceal stored codes, a malicious user can exploit this weakness and be able to recover even strong system passwords. Weak encryption is the most common problem when it comes to securing a system because vulnerabilities are very well known to hackers, so it is easy for them to gain access to a large scale of information structures.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

1.4.2 Encryption of cases (Guessing)

Many passwords can be guessed either by humans or cracking programs supported by dictionaries and personal information of the user. Many users are using weak passwords, usually something about themselves eg name surname date of birth, etc. So it is easy prey for password cracking programs to find the code.

Some users neglect to change the password given to them after buying a computing unit or a connection etc.. Also, some companies neglect to change the default passwords provided by the supplier of operating systems or hardware vendor. One example is the use of Field Service username with the word" Guest "as the password. If this code remains in a computer system then a random user who has dealt with such systems will crack the password. The default list of passwords are available online and can be seen any time by anyone.

The personal information of individuals is now available from various sources, usually from the internet, and often taken by a person who deals with the safety of these (Eg security control). Someone who knows the person can guess the code and see all personal information held by another person. Then he could use a cracking program that will accept information about the user and generate common variations of elements that it needs to find the codes.

1.4.3 Phishing

There is an easy way to hack: Ask the user for his/her password..Phishing attacks nowadays consist of several ways of getting information literally doing nothing. Hackers just create clone sites that ask the users to enter information or even their usernames and passwords. Most users are not security oriented so that they won't realize if it is actually the genuine site or not.

They are directed there mostly by spam mails, for example banks that ask for information validation or other payment information. The result is that once the hackers get the information they can right away compromise the system they want. There is a huge success rate in these attacks because of the unawareness of the users even though the most recent years organizations try to educate people on how to react in these kind of situations.

1.4.4 Social engineering

Social engineering is the latest term when it comes to extracting information in order to gain access to password protected systems. The idea behind this attack is to personally contact the user, employee or the person in charge and pretend to be some security expert trying to solve a security problem.

Everyone will be amazed by the success rate of doing so because people usually don't ask for credentials when they are talking to the phone and when they are in a stressful situation. They very often give the passwords immediately or give very useful information for the people in charge, so it becomes an easy thing for the crackers to breach the security of the system.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

1.4.5 Malware

When referring to malware people understand viruses and worms trying to destroy their computers or do harm. Malware means that of course but when talking about passwords the reference is made for key loggers in general.

Key loggers are some very quiet and mostly not discoverable programs that are installed in computers without the permission or access of the user and record every typing done or take screenshots of login information and then transfer the information gathered to malicious sites or users.

That way it is very easy for the hacker to enter a system and do whatever he pleases with it.

1.4.6 Offline cracking

Many times a system is compromised before with another type of hack like backdoors or Trojans that give most of the times permanent remote access to the malicious user. That said he can take all the time he wants and access the stored hash files and try to crack them in order to find the passwords he needs.

He won’t even get noticed because he will not try to breach the system security, he will just be like an administrator accessing some files. Then he can continue his work offline putting in effect cracking programs and trying to extract the passwords.

1.4.7 Spidering

Modern hackers have to study enough so they are informed for the latest business issues and business news. They have realized that corporate passwords are made up by using business terms as well.

So they study business literature and try to come up with their own wordlists in order to use them in brute force attacks. And they succeed often because businesses are very predictable when it comes to security issues such as passwords.

1.4.8 Brute-Force attack

The simplest but least effective and slowest way to find passwords is Brute - Force attack. It systematically tries all possible combinations of letters, digits and special characters in every possible length until a valid code is found .Of course, faster results can be achieved by combining multiple systems which means using more than one computers or computer clusters. The Brute - Force attack is typically used when the dictionary attack fails because it always finds the desired password but it is computationally expensive as it uses all the computer power to complete each task.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

1.4.9 Dictionary attack

A dictionary attack is an optimization of brute force attack. It actually uses the same procedures as brute force does but the main difference is that it uses a list of words, phrases special characters, common passwords to match the password in demand.

Each word in the file is hashed, and its hash is compared to the password hash. If they match, that word is the password. These dictionary files are constructed by extracting words from large bodies of text, and even from real databases of passwords. Further processing is often applied to dictionary files, such as replacing words with their equivalents ("hello" becomes "h3110"), to make them more effective.

1.4.10 Lookup Tables

Lookup tables are an extremely effective method for cracking many hashes of the same type very quickly. The general idea is to pre-compute the hashes of the passwords in a password dictionary and store them and their corresponding password, in a lookup table data structure. A good implementation of a lookup table can process hundreds of hash lookups per second, even when they contain many billions of hashes.

1.4.11 Reverse Lookup Tables

This attack allows an attacker to apply a dictionary or brute-force attack to many hashes at the same time, without having to pre-compute a lookup table.

First, the attacker creates a lookup table that maps each password hash from the compromised user account database to a list of users who had that hash. The attacker then hashes each password guess and uses the lookup table to get a list of users whose password was the attacker's guess. This attack is especially effective because it is common for many users to have the same password.

1.4.12 Rainbow tables

The Rainbow tables are the new generation in finding passwords (using advanced methods for "cracking") which are encrypted with algorithms, such as the Message Digest 5 (MD5) or Lan Manager (LM). They have become more popular and widely known for speed by which the encrypted with these algorithms passwords can be found.

It is essentially a special type of tables which offer a method based on the combination of time - memory (time-memory trade-off), which is used for finding the unencrypted code from

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

an encrypted one produced by a one-way hash function (the MD5 algorithm eg is an address, which means that it can not be decrypted, only to be seen).

The combination of time – memory technique was issued to reduce the time required for decryption. The idea behind the method is to make all calculations in advance and store the results in folders, called Rainbow tables. The procedure for calculating Rainbow tables takes a reasonable time interval. However as soon as the calculation is over finding passwords with the method mix time - memory is several hundred times faster than the method of exhaustive search.

1.5 Hacks , Cracks

These methods are usually illegal, especially if someone is trying to find a password that is not his. While the Internet offers this kind of software they must be used correctly and with correct understanding by the user. Often, these cracks and downloading various hacks can also be "Trojan horses" or virus, etc.

The difference in these two conditions (cracking and hacking) is not known to the greater portion of the population. In essence the cracker is one which has destructive moods. A hacker does not focus to steal passwords and other information in order to use it against someone.

On the contrary, hacker is someone who collects knowledge about a software, gaining access points that a user does not know or can not have and based in his knowledge he expands it by giving him new opportunities or finding its problems (bugs). On the other hand cracker is someone who illegally gains access, overcoming security systems in order to harm the software or system that has been targeted.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

2.HOW TO MAKE PASSWORD CRACKING HARDER

2.1 Password entropy

Entropy in physics is considered to be a measure of the number of specific ways in which a system may be arranged, often taken to be a measure of disorder.

Entropy has a second meaning when considering information and when talking about information theory in general.

In information theory, entropy is a measure of the uncertainty in a random variable. It was first introduced by Claude Shannon as an approach to measure the amount of information that is unknown due to random variables. It is like attempting to determine the randomness of a variable counting on knowledge contained in the rest of the message.

Now let's move further and apply the entropy term to passwords. It is clear that in order to make passwords harder to crack or sometimes computationally infeasible or not worth to crack, password entropy has to be taken into serious consideration.

To be more specific, when talking about password entropy we refer to the strength that a password has against guessing. That means that apart from the times needed to guess a password with certainty, the base 2 logarithm number is introduced which is the number of bits (entropy bits) in a password. Let's assume that a password with 32 bits of strength calculated the way described before would be as strong as a string of 32 bits chosen in random order. That means that a password with 32 bits of strength would need 2^{32} attempts to exhaust all possibilities during a brute force attack.

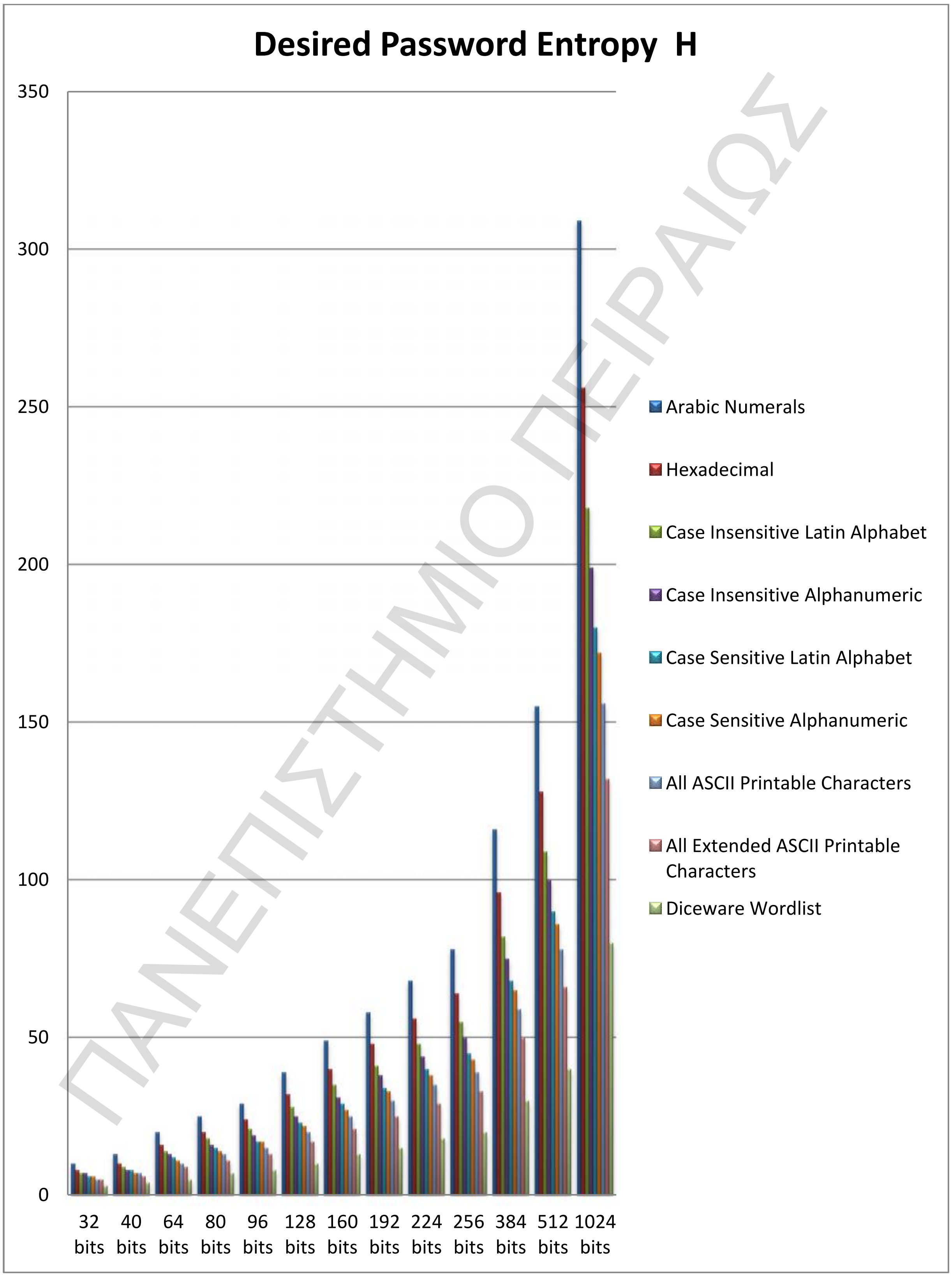
So if a bit of entropy is added, it immediately doubles the number of guesses required and makes the attackers job more difficult. To shape a better picture of that theory a complete graph is being displayed below.

Now, NIST with the special publication 800-63 have made specific suggestions and guidelines as far as password entropy is concerned, in order to approximately estimate the entropy of human generated passwords. Some basic suggestions are: (2)

- The entropy of the first character is four bits;
- The entropy of the next seven characters are two bits per character;
- The ninth through the twentieth character has 1.5 bits of entropy per character;
- Characters 21 and above have one bit of entropy per character.
- A "bonus" of six bits is added if both upper case letters and non-alphabetic characters are used.
- A "bonus" of six bits is added for passwords of length 1 through 19 characters following an extensive dictionary check to ensure the password is not contained within a large dictionary. Passwords of 20 characters or more do not receive this bonus because it is assumed they are pass-phrases consisting of multiple dictionary words.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

As a result, if password strength is in question, as it always will be there are some decent tips to follow in order to achieve better password creation and make crackers life a bit harder.



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Figure 1: Password Entropy Graph

2.2 Key stretching

In cryptography, we meet the term key stretching when referring to methods of making a possible weak key, often password or passphrase more secure against brute force attack by extending the time the attack needs to test each possible key.

Humans tend to create password that are short in length in order to be easily memorized thus making them more predictable. That way they are more vulnerable to password cracking techniques.

Key stretching or key strengthening methods operate like this. (5) The key mentioned before (the weak one) is inserted into an algorithm that runs on a certain speed of processor which takes a standard time to apply. The design of the algorithm introduces a certain amount of delay (typically one second using modern personal computers) that is mostly accepted by users. When the procedure ends the algorithm provides the stretched key. The stretched key must be of a considerable long size (eg 128 bits) so that the whole procedure matters as to make it unbreakable by brute force attacks. The algorithm used should be secure in a manner that there is no room left for diversions as to shorten the time needed for the key calculation.

That said it is up to the cracker to decide what to do. The first option is to try an exhaustive search on the new key which is computationally infeasible when the length of the key is significant or try different combinations of the first key. If the first key before the algorithm digestion is a password or passphrase the cracker has to search every word on a dictionary or common passwords list and afterwards check all possible combinations of longer passwords. Key stretching techniques cannot avoid this from happening but they can really slow down the attacker who has to spend a great amount of time.

Even if the cracker has the same hardware or better from the user he still has to spend a lot of time as he has to check every possible combination whereas all the user has to do is just use the algorithm and produce the new key.

Key stretching can be carried out in many ways. One example is that a hash function or a block cipher could be applied again and again in a loop. To better understand that way a table below is formed with the code. (5)

```
Key = “ ”

For 1 to 65536 do

    Key = hash (key + password )
```

Figure 2: Key Stretching Code

A similar technique called salting which protects from time-memory tradeoff attacks could also supplement the above code to make the derived key even more stronger.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

2.3 Salting

Salt ,in cryptography is called the random data that are inserted through a hash function and have as an outcome a new key (password or passphrase). (4)

Each password has to have a new salt. Most times the salt and the password are concatenated and inserted to the cryptographic hash function and the output is stored with the salt in a database. The role of hashing is to prevent possible attacks of the plaintext passwords when the database is no longer safe and call for authentication.

Salting first use was to block precomputation attacks like rainbow tables which are used for making password cracking very fast and efficient. Salting intends to make passwords stronger by making them longer. That way it is harder to crack because it is well known that every additional character added to a password exponentially increases the amount of possible character combinations.

Once the attacker finds out what the salt is it is no longer useful because it is just like it was never there. He can just edit the program and the code he uses for cracking add the salt and try again to break the passwords. Most of the times he succeeds.

So the question now is what salting technique is better. Random salts as mentioned before or unique salts? Unique salts mean the one that every user knows only for himself like his birthday or email. The thing is that an answer is never clear to that question. It's all a matter of how salts are stored and how safe the storing is. If both salts are used and stored in the same database as the password it's the same thing because once the attacker gains access he will find out what the salt is and the game is over. So a good practice would be to separate the databases from which the salting and the passwords are stored. In that situation it is better to use random salts because the attacker may have compromised the database with the hashed passwords which could include usernames emails, so a random salt is hard to be guessed.

Moreover, for even greater security adding to the fact that the salt is stored in that database, it could be added to the actual source code of the login/register script. So the attacker would definitely have to gain access to both the source code and the database to find out what the salt might be.

That said, it becomes clear that salting has a lot of benefits in the race of defending passwords from possible attacks. The main and the most serious attack like rainbow tables which are precomputed hash files can be blocked fairly enough because the precomputation doesn't include the salt so it is impossible to find the passwords in demand. That is because salts extend the complexity of the password, for example a 12 byte password and a 2 byte salt makes a 14 bytes password. (4)

Finally it is important to mention that salting slows down also dictionary and brute force attacks when it comes to large amounts of passwords. This means that every password will possibly have different salt so each guess should be separately hashed for each salt which makes the procedure quite slow. Even if users wanted to use the same passwords to more than one account, adding salt would result to more than one different passwords in the end of the procedure. No one then would be capable of uncovering that they are the same just by reading the hash files.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

2.4 Time needed to crack passwords

The time required to 'crack' a password is directly related to the strength of that password. Most methods of Password Cracking require a computer to produce many different codes for the one needed and constantly applying each one of them. A typical example is the Brute-Force technique in which a computer tries every possible combination of keys or letters until it succeeds.

Most common Password Cracking Techniques such as the Brute-Force attack, Dictionary attack, Hybrid attack try to reduce the number of tests needed to find a password. Of course a complex code results in the exponential increase of the number of passwords the attacker needs to try. So the likelihood of finding this code using any of the above techniques becomes very difficult.

Approximately though, how much time is required by a computer or better by a cluster of computers to guess various passwords? The data shown below are approximate and are the maximum time required to find passwords using a simple case of Brute-Force attack.

2.5 Classes of Attack

Below are six classes according to the speed of breaking passwords. These are:

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

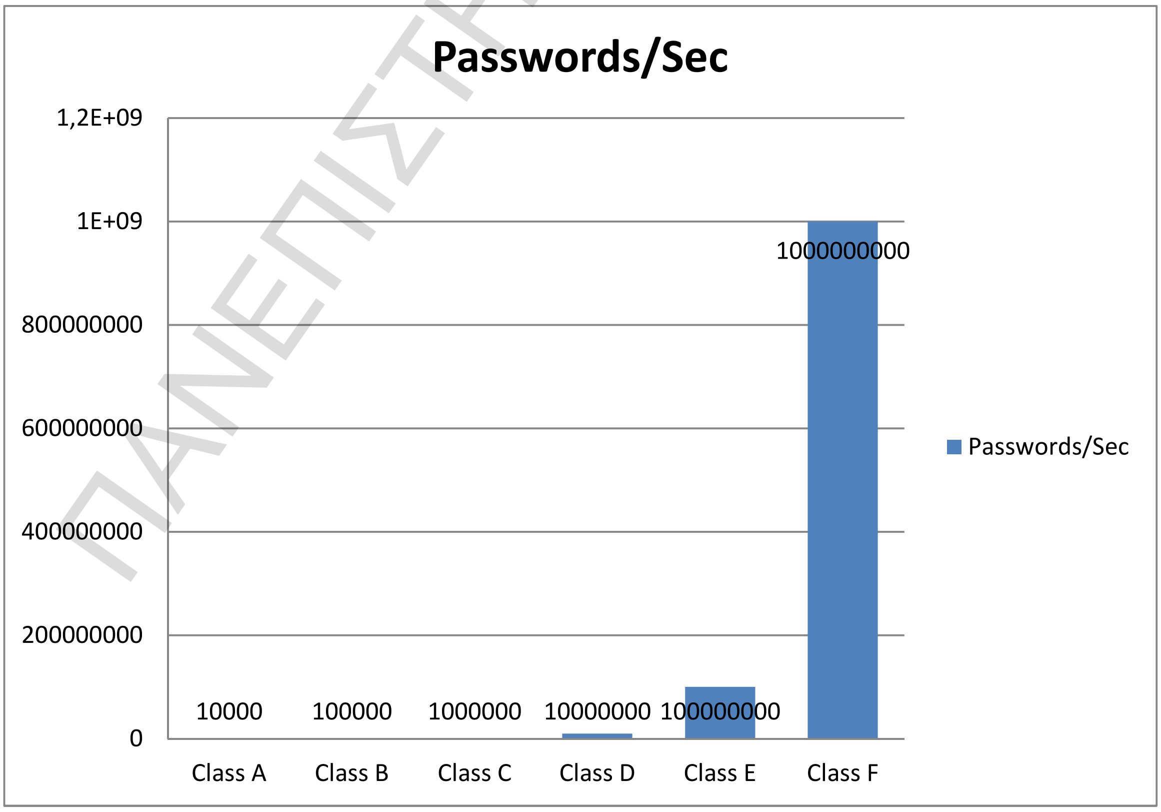


Figure 3: Classes of Attack

Below are some tables that give a pretty good picture of why the password length do matters when it comes to the cracking process.

10 Characters

Numbers. As one can see, choosing a password from a small range of characters is a bad idea.

Numbers	0123456789						
Password		Class Of Attack					
Length	Combinations	ClassA	ClassB	ClassC	ClassD	ClassE	ClassF
2	100	Instant	Instant	Instant	Instant	Instant	Instant
3	1000	Instant	Instant	Instant	Instant	Instant	Instant
4	10000	Instant	Instant	Instant	Instant	Instant	Instant
5	100000	10 Sec	Instant	Instant	Instant	Instant	Instant
6	1Million	$1^{1/2}$Mins	10 Sec	Instant	Instant	Instant	Instant
7	10Million	17 Mins	$1^{1/2}$Mins	$1^{1/2}$Mins	Instant	Instant	Instant
8	100Million	$2^{3/4}$Hours	17 Mins	$1^{1/2}$Mins	10 Sec	Instant	Instant
9	1000Million	28 Hours	$2^{3/4}$Hours	17 Mins	$1^{1/2}$Mins	10 Sec	Instant

Figure 4: 10 Characters

26 Characters

The full Latin alphabet, either uppercase or lowercase (not both in this case).

Uppercase	ABCDEFGHIJKLMNOPQRSTUVWX						
Lowercase	abcdefghijklmnopqrstuvwx						
Password		Class of Attack					
Length	Combinations	ClassA	ClassB	ClassC	ClassD	ClassE	ClassF
2	676	Instant	Instant	Instant	Instant	Instant	Instant
3	17576	< 2 Sec	Instant	Instant	Instant	Instant	Instant
4	456976	46 Sec	5 Sec	Instant	Instant	Instant	Instant
5	11.8Million	20 Mins	2 Mins	12 Sec	Instant	Instant	Instant
6	308.9Million	$8^{1/2}$Hours	$51^{1/2}$Mins	5 Mins	30 Sec	3 Sec	Instant
7	8Billion	9 Days	22 Hours	$2^{1/4}$Hours	13 Mins	$1^{1/2}$Mins	8 Sec
8	200Billion	242 Days	24 Days	$2^{1/2}$Days	348 Mins	35 Mins	$3^{1/2}$Mins
9	5.4Trillion	17 Years	21 Mon.	63 Days	$6^{1/4}$Days	15 Hour	$1^{1/2}$Hours
10	141Trillion	447 Years	45 Years	$4^{1/2}$Years	163 Days	16 Days	$39^{1/4}$Hour
12	95Quadrillion	302.603Years	30.260 Y	3.026 Y	302 Years	30 Year	3 Year
15	1.6Sextillion	53TrillionY	532Mil.Y	53 MillY	5 MillY	531.855 Y	53.185 Y
20	19.9Octillion	63 Quadr. Y	6.3 QuaY	631TriY	63.1TriY	6.3 Tri. Y	631 BilY

Figure 5: 26 Characters

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

36 Characters

The full Latin alphabet, either uppercase or lowercase (not both in this case) and the numbers.

Uppercase	ABCDEFGHIJKLMNOPQRSTUVWXY
Lowercase	abcdefghijklmnopqrstuvwxy
Numbers	012345678

Password		Class of Attack					
Length	Combinations	ClassA	ClassB	ClassC	ClassD	ClassE	ClassF
2	1296	Instant	Instant	Instant	Instant	Instant	Instant
3	46656	4 Sec	Instant	Instant	Instant	Instant	Instant
4	1,6Million	$2^{1/2}$Mins	16 Sec	$1^{1/2}$Sec	Instant	Instant	Instant
5	60.4Million	$1^{1/2}$Hours	10 Mins	1 Mins	Instant	Instant	Instant

Figure 6: 36 Characters

52 Characters

This time the whole alphabet, using a mix of uppercase and lowercase letters, which essentially doubles the number of combinations, compared with only a single case (eg Uppercase or lowercase).

Upper-Lower Case	AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz

Password		Class Of Attack					
Length	Combinations	ClassA	ClassB	ClassC	ClassD	ClassE	ClassF
2	2704	Instant	Instant	Instant	Instant	Instant	Instant
3	140608	14 Sec	<2Sec	Instant	Instant	Instant	Instant
4	7.3Million	$12^{1/2}$Mins	$1^{1/4}$Mins	8Sec	Instant	Instant	Instant
5	380Million	$10^{1/2}$Hours	1 Hour	6 Mins	38Sec	4Sec	Instant
6	19Billion	23 Days	$2^{1/4}$Days	$5^{1/2}$Hours	33 Mins	$3^{1/4}$Mins	19Sec
7	1Trillion	$3^{1/4}$Years	119 Days	12 Days	$28^{1/2}$Hour	3 Hours	17 Mins
8	53Trillion	$169^{1/2}$Years	17 Years	$1^{1/2}$Years	62 Days	6 Days	15 Hours
9	2.7Quadrillion	8.815 Years	881 Years	88 Years	9 Years	322 Days	32 Days

Figure 7: 52 Characters

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

62 Characters

Combination of numbers, upper and lowercase letters.

Upper- Lower case Numbers		0123456789AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz					
Password		**Class of Attack**					
Length	**Combinations**	**ClassA**	**ClassB**	**ClassC**	**ClassD**	**ClassE**	**ClassF**
2	3844	Instant	Instant	Instant	Instant	Instant	Instant
3	238328	23 Sec	<3Sec	Instant	Instant	Instant	Instant
4	15Million	24½ Mins	2½ Mins	15 Sec	< 2 Sec	Instant	Instant
5	916Million	1 Days	2½ Hours	15¼ Mins	1½ Mins	9Sec	Instant
6	57Billion	66 Days	6½ Days	16 Hours	1½ Hours	9½ Mins	56Sec
7	3.5Trillion	11 Years	1 Year	41 Days	4 Days	10 Hours	58 Mins
8	218Trillion	692 Years	69¼ Year	7 Years	253 Days	25¼ Days	60½ Hour

Figure 8 : 62 Characters

86 Characters

Combination of uppercase and lowercase letters and special characters.

Upper-Lowercase Special Characters		AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZ <SP>!"#$%&'()*+,- ./:;<=>?@[\]^_'{|}~					
Password		**Class Of Attack**					
Length	**Combinations**	**ClassA**	**ClassB**	**ClassC**	**ClassD**	**ClassE**	**ClassF**
2	3844	Instant	Instant	Instant	Instant	Instant	Instant
8	218Trillion	9.488 Years	948 Years	94 Years	57 Years	346 Days	34 Days

Figure 9: 86 Characters

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

96 Characters

.
Combination of uppercase and lowercase letters, numbers and special characters.

Upper-Lower Case Special Characters Numbers	0123456789AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUu VvWwXxYyZz<SP>!"#$%&'()*+,./:;<=>?@[\]^_‘{|}~						
Password		**Class Of Attack**					
Length	**Combinations**	**ClassA**	**ClassB**	**ClassC**	**ClassD**	**ClassE**	**ClassF**
2	9216	Instant	Instant	Instant	Instant	Instant	Instant
3	884736	88½ Secs	9 Sec	Instant	Instant	Instant	Instant
4	85Million	2¼ Hours	14 Mins	1½ Mins	8½ Secs	Instant	Instant
5	8Billion	9½ Days	22½ Hour	2¼ Hours	13½ Mins	1¼ Mins	8 Sec
6	782Billion	2½ Years	90 Days	9 Days	22 Hours	2 Hours	13 Mins
7	75Trillion	238 Years	24 Year	2½ Years	87 Days	8½ Days	20 Hours
8	7.2Quadrillion	22.875 Year	2.287 Y	229Years	23 Years	2¼ Years	83½ Days

Figure 10: 96 Characters

2.6 Password creation

There are many ways of creating a password. The most commonly used are password creators that can be found online and there is a plethora of them to choose. The way they work is to randomly process the data entered by the user and generate password after giving specific guidelines and rules to the user as for the password strength. Then the user can memorize that and use it wherever he wants. The sad thing about this procedure is that the randomly generated passwords are not that random because they can be precisely figured out by brute force attacks.

So this leaves no option to the user if he wants to have a secure password, to make own on his own. This usually is carried out after taking consideration suggestions and rules that come across the user when he wants for example to open a new email account. The service estimates the strength of the password entered and informs the user by giving him visual recreations of green yellow and red bars. These services have been programmed to know weak passwords and tell the user what is better for him even suggesting alternatives for his password.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

2.7 Password policy

Let’s move on now to something really crucial when considering information and better password safety.

Password policies are the result of an effort to force users to make proper use and generate strong passwords. (3) They are found often in corporations and organizations as a part of security oriented awareness projects targeted to all users from the owner to the last employee. If applied correctly they can have profound results in securing the organization from potential security incidents. They can be an extra measure taken against those who intend to do harm.

These policies mainly consist of four parts. First, the password length and format. Second, the password duration. Third part is common password practice and last the actions taken when alleviated from the policy. (3)

A lot of policies suggest certain password length (eight characters minimum) in order to comply with the needs of the service or organization. As far as the format is concerned various policies can be found but the most common are summarized to the use of both uppercase and lowercase letters, use of numbers, use of special characters,and not the use of personal information such as emails, usernames, private information such as plate numbers and so on. Some systems though just don’t leave a choice for the user to choose the password and they provide a number of preselected ones.

Password duration is a critical aspect of the policy also because it means that the less a password is active the less chances there might be for an attacker to try and break it. So organizations worldwide have a predefined number of time that user passwords stay the same and when that ends they are forced to change their passwords. Because humans are involved in this process and because the time the password remains the same can be really short, a better practice would be to demand a very strong and secure password instead of changing it in a three months time for example.

Password policies also suggest safety features for users to apply and manage their passwords properly. Common practice could likely be the denial of sharing accounts and use of same passwords to more than one account, avoiding phishing efforts and to never write down these information. Also users are told to log off from their computers before leaving their desk and to immediately change a password if there is suspicion that someone already knows it.

Last but not least, because of the human nature and the limited memory many of these policies remain unapplied leaving the organization or service literally unprotected to the world even if extreme technical security measures have been taken to avoid such occurrences. Taking into consideration the damage caused in the past from such occasions of security breaches, password policies clearly state the actions taken to punish the one that doesn’t apply them. It is easy then for the organization to cope with the person or people who did forget what to do.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

2.8 Guidelines for creating strong passwords

It is time to present some valuable information about how to make strong passwords and suggest some common guidelines in order to be as protected as anyone can be from password cracking. (2)

Password Length	Minimum Of 12-14 characters
Password Characters	Uppercase, Lowercase, Numeric, Special Characters
Password Format	No use of dictionary words,usernames,private information like birthday,plate number,street number
Password Usage	Different Passwords for Each Service
	Never Write Down Passwords
	Immediately Change Default System passwords
	Never Share Passwords With Others

Figure 11: Guidelines for strong passwords

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

3.USING PASSWORD CRACKING TOOLS

3.1 How to make password cracking faster

3.1.1 GPU's

''A password-cracking expert has unveiled a computer cluster that can cycle through as many as 350 billion guesses per second. It's an almost unprecedented speed that can try every possible Windows passcode in the typical enterprise in less than six hours.'' (6)

That said everyone can just imagine what modern computers can do in term of cracking a password. When talking about computers we have to be more specific and make special reference to GPU's. They have been in the computing scene a long time ago and everyone has associated them with the graphics performance of a system. A GPU is excellent at processing mathematical calculations. Graphics rendering is simply a series of complex mathematical calculations. So are hashing algorithms.

A GPU has hundreds of cores that can be used to compute mathematical functions in parallel. A CPU usually has 4-8 cores. Although a CPU core is much faster than a GPU core, password hashing is one of the functions that can be done in parallel very easily. This is what gives GPUs a massive edge in cracking passwords.

As a result it is given the ability to test multiple hashes in once and get the passwords much quicker. Modern password cracking tools make use of graphic cards that in a middle ranged computer system can have up to 2 gb of ram and multiple cores as mentioned above. So it is a hackers best practice to use multiple GPU's to have the results he is looking for.

3.1.2 Computer clusters

Now let's move on to something even bigger than graphics. When referring to computer clusters it is more than clear that it means multiple computer systems interconnected working as one. They usually are stored in large rooms even buildings and are connected to each other mostly through fast LAN, when each computer of the chain is running one instance of a software.

It needs no special skills though to realize what these systems can achieve when it comes to password cracking. The speed, the multiple cpu's and gpu's, the billion calculations per second and many more that they offer can certainly be a weapon of mass destruction if found in the wrong hands.

Malicious users nowadays can have access to these systems by having compromised them before with other hacks or by creating bot computers throughout the web all over the world. So they don't have to actually own any of these systems. All they need to do is summarize their skills and gain access to one of them. Then it is a matter of what they want to do, what evil thoughts they have in mind and what are their goals.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

The success rate for each hacker ranged from 62% to 90%, and the hacker who cracked 90% of hashed passwords did so in less than an hour using a computer cluster.

3.2 Most Used Programms

Below some programs which are used for "breaking" passwords will be demonstrated. There are too many kinds of such programs and if someone searched through the internet he will be astonished by how vulnerable his computer and passwords could get. A table of comparison was conducted between the most known and used tools (Cain & Abel, Jtr, Passwords Pro, Hashcat, Rainbowcrack, Ophcrack).

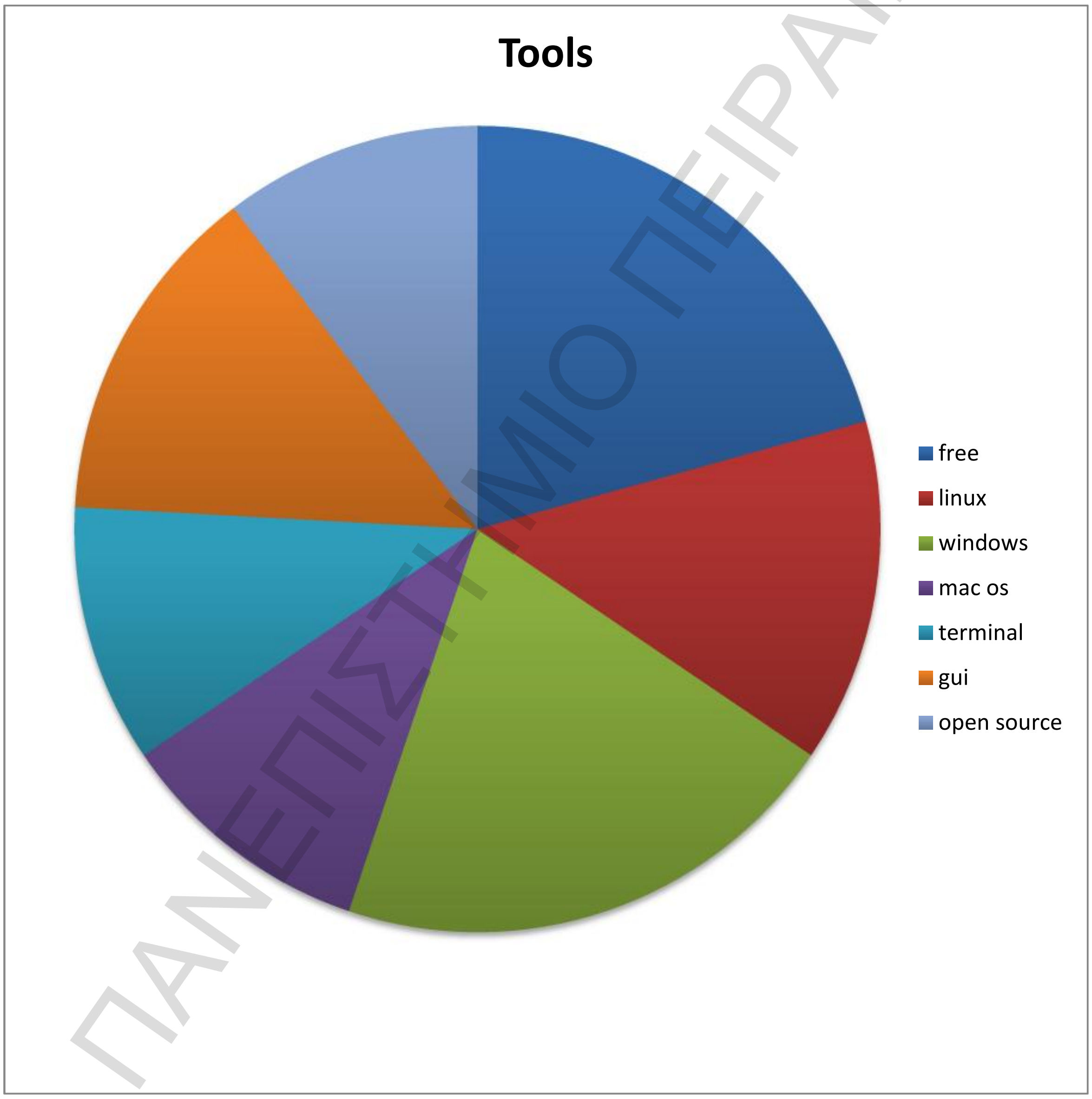


Figure 12 :Tools Comparison Pie

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

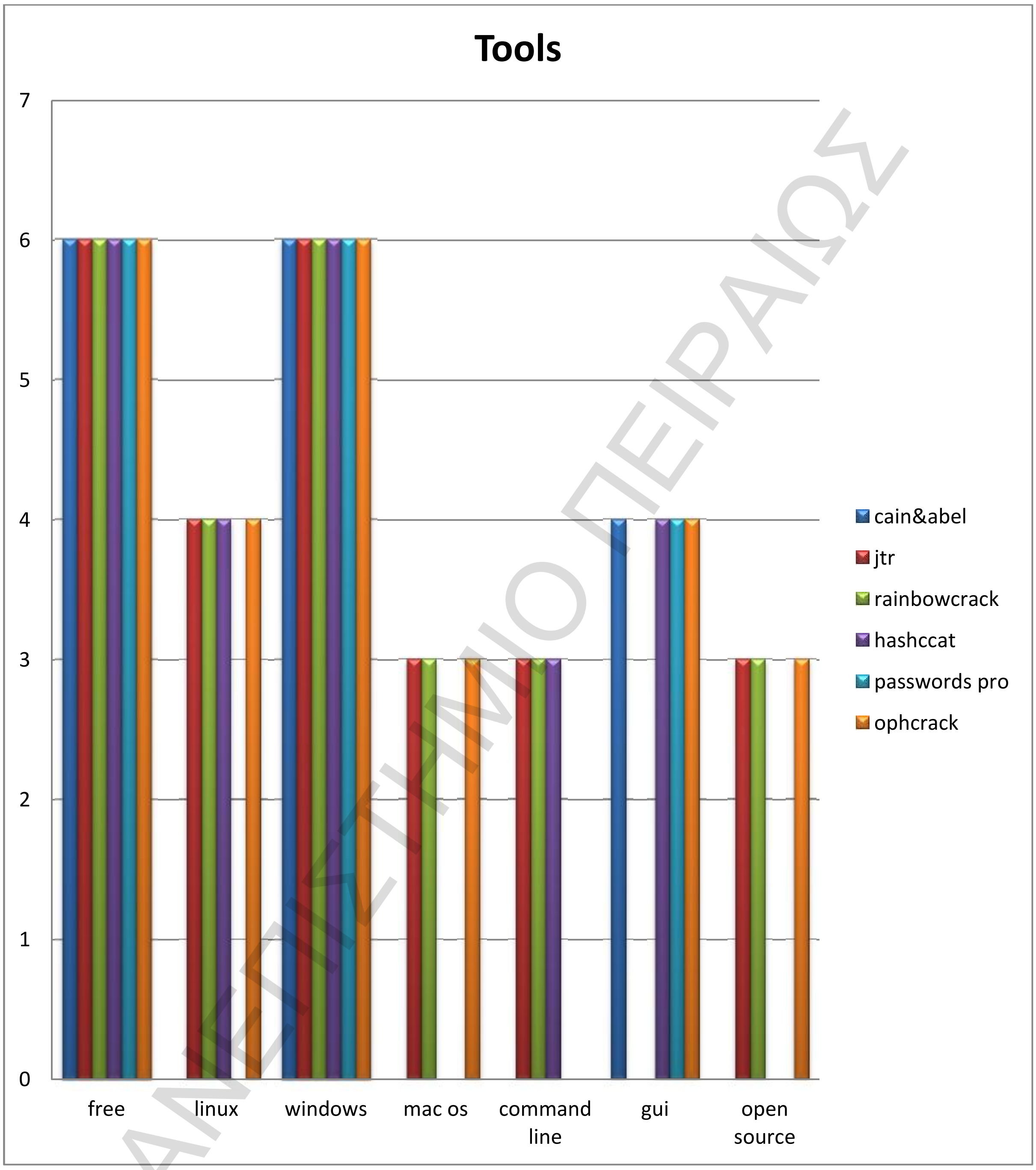


Figure 13:Tools Comparison Chart

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

3.2.1 CAIN & ABEL

The Cain & Abel is a password recovery tool for operating systems of Microsoft. It can recover many kinds of passwords using methods such as dictionary attacks, brute force and cryptanalysis attacks, recording conversations VOIP, recovering wireless network keys, revealing the stored passwords and analyzing routing protocols.

Cain & Abel has been developed in the hope that it will be useful for network administrator, teachers, security consultants, software vendors and security for anyone who wants to use it for ethical purposes. The program warns that there is the potential to cause damage or loss of data and that the user is solely responsible. (12)

<u>Types of attacks</u>

- ***Brute force attack***: The brute-force attack refers to the exhaustive testing of possible keys that produce a cipher text to reveal the original message. Such attacks, which use all possible keys, can always be made. Often, however, the attacker launches the attack using more "potential", in his view, trying in this way to find the key faster. In practice, the search will stop when the key is found, without needing further update to the list of keys.

- ***Dictionary attack***: The dictionary attack uses the brute force technique of trying all the words in an exhaustive list (predefined list of values). Unlike the brute-force, dictionary attack tries only those features that are most likely to succeed. In general, the attacks are successful because many people tend to choose passwords that are small (7 or fewer characters), single words found in dictionaries or simple, easily-variants of words.

- ***Cryptanalysis attack***: This tactic is quite fast but it is useful when breaking only a few types of encrypted passwords. It uses a set of large tables of pre-calculated encrypted passwords (Rainbow Tables), to improve its exchange methods which are known today and recover faster various codes.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

3.2.2 RAINBOW CRACK

A wide variety of software producing Rainbow tables and crackers using this method is found nowadays. The mostly used is Rainbow crack..

Before beginning the process it is necessary to have available a multitude of Rainbow tables. The amount of space that the user wants to have allocated depends solely on his preferences and needs. If available less than 100GB it is advisable not to load large number of these.

The program used to produce them is "rtgen". To produce the Rainbow tables certain parameters must be specified as shown below: (11)

Hash algorithm options:	(exp: md5)
Character set:	(exp: alpha-numeric)
Plaintext length minimum:	(exp: 1)
Plaintext length maximum:	(exp: 8)
Rainbow Table Index:	(exp: 0)
Rainbow Chain Length:	(exp: 11300)
Rainbow Chain Count:	(exp:6000)
File Title Suffix:	(exp: 0)

Figure 14: Rainbow Crack Parameters

After calculating the period which will run the process of creating them the process production can be initiated. The disheartening is that it usually takes too much time.

Then it has to be classified with "rtsort", which all it does is to organize the newly produced Rainbow table. Finally the Rainbow table is ready to "crack" hash values and return passwords with the help of the program "rcrack", a subdirectory which stores all Rainbow tables.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

3.2.3 OPHCRACK

The Ophcrack is an open source program, which is able to find the passwords in a system ,using LM Hashes through rainbow tables. It has Graphical User Interface and runs on Windows, Linux and MAC OS. The program can extract the hashes of a computer and more specifically the sam files which contain the codes of the system. Can crack 99.9% codes that consist of letters and numbers and are up to 14 characters in a few minutes.

The Rainbow Tables for LM Hashes passwords which consist of letters and numbers are free from the creators of Ophcrack. The LM hash (LAN Manager) is one of the format of Microsoft Lan Manager and Windows which is used to store users passwords less than 15 characters. This format is uniquely used in Windows Lan Manager. It can crack easily due to two reasons. The first is that passwords that are more than 7 characters are divided into two pieces and each piece has a different hash. The second reason is that all small letters containing the code are converted into large before coded.

The Ophcrack first used rainbow tables in 2003, a process which aimed at weaknesses coding of LM. Later other cracking tools adopted this idea resulting in quickly finding passwords even within a few seconds. The Live CD can be found at: http://ophcrack.sourceforge.net . (9)

3.2.4 PASSWORDS PRO

This program is designed for recovering passwords out of hashes, and supports over 180 algorithms. The software supports several kinds of attacks which are shown below. (7)

- ***Preliminary Attack*** – this is a quick check of user hashes for matching to simple passwords like "123", "qwerty", "99999", etc., as well as to passwords found by the program earlier.

- ***Brute Force Attack*** – this is the exhaustive search through all possible passwords in a certain range; e.g., "aaaaaa"..."zzzzzz".

- ***Mask Attack*** – this attack is used when some information on the lost password is known. To use the attack, make sure to specify the mask for each character in the password to be recovered in the attack settings. For mask characters the conventional characters for the standard or custom character sets can be used – ?u, ?d, ?2, etc.

- ***Simple Dictionary Attack*** – during this attack, the program simply checks hashes against passwords in dictionaries.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

- ***Combined Dictionary Attack* –** during this attack, passwords are made of several words taken from different dictionaries. That allows to recover complex passwords like "superadmin", "admin*admin", etc.

- ***Hybrid Dictionary Attack* –** this attack allows modifying passwords taken from dictionaries (for example, shift the password to upper case, append '1' to the end of the password, etc.) and validating them as user passwords. The actions performed over source passwords are called "rules", and the full list of those is available in the file "Rules.txt" in the software distributive.

- ***Rainbow Attack* –** this attack attempts to recover passwords using the pre-calculated Rainbow tables. The software includes the following plugins (to obtain more information, see the file "ReadMe.txt" in the folder containing each plugin):

- ***Dictionary Generator* –** generates dictionaries of passwords from a specified range and performs other functions related to using dictionaries – sorting, merging to one file, etc.

- ***Hash Bruteforcing History* –** codes and decodes history of hash bruteforcing.

- ***Hash Generator* –** generates hashes of all types loaded in the program.

- ***Hash Queue* –** handles queues of hashes downloaded from the Internet.

- ***Hidden Passwords Recovery* –** recovers text hidden behind asterisks.

- ***NTLM Password Finder* –** attempts to find the NTLM password on the Passwords Pro hash list with a known LM password by checking it in all possible character cases.

- ***Password Generator* –** generates random passwords with specified parameters.

- ***Password Sender* –** sends recovered passwords to websites.

- ***SQL Dump Parser* –** extracts hashes from SQL dumps of various forums.

- ***Text Converter* –** converts text from Base64 to plain text and the other way around.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

3.2.5 JTR (John The Ripper)

JTR (John The Ripper) is a free program for ’cracking’ passwords. Initially it was developed for the UNIX operating system and currently operates in fifteen (15) different platforms (for eleven specific architectures UNIX, DOS, WIN32, BeOS, and OpenVMS). It can be used to control different encrypted access passwords and mainly aimed at UNIX. To download the program visit the following address: http://www.openwall.com/john/179

The information on accounts and passwords on UNIX are commonly found in the file / etc / passwd. On some systems, this file is accessible to all users (bad practice by the administrator).

A very interesting feature of John The Ripper are the rules , with which you can replace a with 0, i with 1, a with @ and so on. So you could give the program a dictionary that includes words with special characters where JTR with the appropriate rules, will convert the words that include special characters in standard words dictionary. (8)

JTR supports four different methods of cracking:

• **Wordlist mode**: This is the simplest way of cracking with John the Ripper. All it needs to be done is to specify a list of words (a text file containing one word per line). In this method rules of deformating words can be enabled (used to modify or ’tag’ words that produce other possible passwords). If activated, all rules will be applied to each line in producing many words ’possible’ passwords from each source word.

• **Single crack mode**: This method is usually recommended for weak passwords because it includes only a few rules (login names, full names fields or home directories names) and a short list of words to find passwords. The Single crack mode is relatively simple to use and faster than wordlist mode but does not guarantee the result set.

• **Incremental mode**: Considered the most powerful method for cracking ,it attempts to combine all the possible combinations of characters, such as passwords. However, using this method it is possible to never terminate, due to ratio of the number of combinations that is forced to run (in fact will terminate if you set a low threshold if length of the code where it wants to break it or use a small set of characters). To use this method, then it should be set a specific number of parameters such as the length of the password and only check numbers, characters, symbols or a combination of all the previous ones.

• **External mode**: This method is a bit complicated, but to use it skills are needed with John the Ripper. To set the external mode, a configuration file must be created, a section named [List.External: MODE], where the MODE is the name this function was given. This section of the configuration file contains some functions in language C, which the JTR will compile and will use them when called upon and activated it through the command line, with the name given (MODE).

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

3.2.6 HASHCAT

There are also other applications that are almost the same with the above mentioned. Some of the them are custom made by the programmers some use custom scripts and some combine all the methods described previously. It is worth to mention though that one very useful tool in the password cracking field is the hashcat tool.

All it does is to crack hashes provided by using rules and dictionaries just like john the ripper. But it is very efficient and quick and it can be used along with john the ripper for better results. Hashcat is the world's fastest CPU or GPU based password recovery tool. It was created in the year 2009. It provided additional support for modern multi core CPUs that were absent in the famous "John the Ripper (JTR)" and "PasswordsPro" Softwares. Hashcat comes in three formats to suit the needs of different hardwares. For example, Hashcat is used in the 32bit and 64bit based systems. While the Ocl-Hashcat is use for the GPU based systems to take the advantage of the parallel architecture of the GPU. Another version is Ocl-Hashcat-lite is a subset of the Ocl-Hashcat.GUI based version of all three versions is also available. (10)

Hashcat is a free software and can be obtained from the http://hashcat.net/hashcat/ while the hashcat-gui can be download from the http://hashcat.net/hashcat-gui/. Hashcat-linux GUI is a little different in display as compared to the windows counterpart.

Hashcat supports the following attack modes:

0 = Straight – It runs all words in a dictionary against a given hashlist.

1 = Combination – Combines words from the given dictionary.

2 = Toggle-Case – Flips all capitals to lowercase, and all lowercase to uppercase. Digits and special characters are ignored as they do not have a case.

3 = Brute-Force – BF should not be used because it is not effective against long passwords. It takes lots of time to succeed.

4 = Permutation – Takes letters from a word, and re-arranges them.

5 = Table-Lookup – Breaks a string into individual characters and applies a rule to each one matching a table entry. A folder name tables/ containing tables is provided with the distribution.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

4. CONTEST 2012 (PASSWORD CRACKING)

As a part of this dissertation it was asked to analyze the 2012 contest of Korelogic CMIFC (Crack Me If You Can) and try to crack as many hashes as possible and present results from the procedure and the tools used. The hash files were obtained from the https://contest-2012.korelogic.com/ web site. Wordlist files were downloaded from the following websites

1	MD5 Decrypter Passwords [50.6MB / 181MB] Updated: 19-Feb-2013 It is a passwords list that always proves its worth. It is compact, clean and it always yields results. It is an example of how an ideal wordlist should be. It is the default wordlist in the project in hand. http://www.md5decrypter.co.uk/downloads.aspx
2	MD5Decrypter Output Wordlist [144MB / 511MB] It is the second wordlist used. It is also yields results in a timely manner. http://www.md5decrypter.co.uk/downloads.aspx
3	http://contest-2010.korelogic.com/wordlists/wikipedia-wordlist-sraveau-20090325-sorted-shrunk.dic.gz [65.9MB / 224.9MB] It is the third wordlist most used. It is quite useful but it may not prove it's worth as compared to the previous wordlist.
4	naxxatoe-dict-total-new-unsorted [1.13GB / 31.1GB] It is the fourth one used a little. It is an example of a bad dictionary. It is unsorted, unclean and always time consuming. http://bitsnoop.com/naxxatoe-dict-total-new-unsorted-q5403423.html

Figure 15: Wordlists

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Hash files were trimmed and cleaned according to the specifications provided by the hashcat team. http://hashcat.net/wiki/doku.php?id=example_hashes...That was possible using the following website http://www.md5decryptor.co.uk ..It trims the hashes according to the hashcat specifications just by providing the hash file online. During the cracking process there has been an extensive use of Hashcat version because it was extensively used by the winners of the contest to recover the passwords against the given hashes. It has to be mentioned though that to shape a better picture of what cracking tools can do and make comparison and suggestion other tools like JTR was also implemented. To save space only the first 15 recovered hashes are shown from each procedure. Below there will be a brief technical presentation of the cracking process and the hashes recovered.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

4.1 Hashcat

4.1.1 MD4 Hashes

```
./hashcat-cli32.bin –m 900 –remove myhashes-9-raw-md4_new.txt
dictionary/passwords.txt –r rules/combinator.rule –o
resultscombinatorrule_passwddb2.txt
```

```
1.  9936e4589306b1db3c1e27b6a9001836:elzie2007
2.  1514f6e058830a3c8a68eca58a2e80fe:about time
3.  98ae14f3a1516bebbfb3030d913faed1:kiron2007
4.  e2106919a3c5c60ddb83a1abdc53ac9c:going there
5.  9cf8cca1d04e3c9439f05c2396767b33:unister#2008
6.  1b3d495cb9efdccafc72cd9b94e52ed6:use it
```

Figure 16:MD4 Hashes

```
./hashcat-cli32.bin –m 900 –remove myhashes-9-raw-md4_new.txt
dictionary/passwords.txt –r rules/specific.rule –o specificruleout.txt
```

```
1.  0638267f4eb7372bdf01cdbeef5d004d:!838Deserved
2.  5af79945e930318d57a45b5e44badbbe:#52greystone
3.  22c6ecf48d5ef4bfe28dfdf03c4f92b0:251255807
4.  9b1778977939b8f690f49a5d07a828cd:1223899953
5.  1bcda150e89395af7dc11d06ef8732bb:123456lmn
6.  a899bec156a121842dae42ed4d7447b5:1v1c@k0k5r1c
7.  007aeff8ab36b189a8f9fc63c32ea092:Eastern2976!
```

Figure 17: MD4 Hashes 2

```
./hashcat-cli32.bin –m 900 –remove myhashes-9-raw-md4_new.txt
dictionary/passwords.txt –r rules/generated.rule –o generatedruleout.txt

1.  b0fe207a4b0495189bd06ee92835fa65:p4$$w0rd*
2.  0b4fe1567765a59ef1fff80d754d621f:1022frog1
3.  414b2807f360388ac7f3e9dd5a4a9e3a:$corpion1991
4.  72653967f5e8d59d47115ba269210d1e:@#linduska84
5.  aca83f20056cd5e0f0f67ef8cedc5017:368104186
6.  33d51249c4ee4d2425d161be735bf1e7:f.martino
7.  19bdea71c366205f8d5263d273197db7:abcabc143!@#
8.  fb546c45ae4a89bd731bcf430d80f8c5:211980274
9.  3f17705911b0e892b3fb01bdd92f4bc6:501044284
10. eed9dc7eec098447f7ec75f93e2b0105:Aarwasngen
11. a475e0676b97d0d36ecfd9eec67f3500:027136364
12. 3174dd55000f49b5eb20a6f035e66f03:across the
13. 8218d829816950cafb681b05d68cbec2:270ndrine
14. 3cbb0fff2cc415584e70f28f1bfe6c98:0109218263
15. 21419bd9a19ba2943a0853f41f21b4c2:1634qwer!@#$
```

Figure 18: MD4 Hashes 3

```
./hashcat-cli32.bin –m 900 –remove myhashes-9-raw-md4_new.txt
dictionary/output.txt –r rules/best64.rule –o resultsbest64_outputddb1.txt

1.  8909202ab54bbddf57e244c930e40902:044019900
2.  e8ee9b4d54f9771ceb5e10d08a5202d6:05051988a
3.  5d07c92186344c85585ad18465a3d2fe:199507969
4.  e24cdadcb6d0156a795851d34d31a7aa:21.12.1974
5.  12b12b0f8b5d3e4c2062f0b9e6370578:425378967
6.  69c3369ccb644d1b75869c41ed19336b:305987216
7.  1279de4c9752a68683d3929b62f69ec1:8448tomps
8.  96158ad6f0bde81849b293474b41ee16:4keerkaas
9.  4f88cdab8bb9345478df40c72a7de3c7:Allison25
10. a319a7dda966d80db82828ac9d06910d:cHEMIStRY
11. 0af07649a4aa0407028d3fe65c611446:CHISHOLM3
12. a13fb2f88302d29b8c8d08283a4412c4:Deinstag4
13. 0db33dcf25358d9ae0e485579b49410e:ssentsriF
14. 14f5fe2b695492217ea6b8c1d71fdae0:Caborojo6
15. 406045a7c7fa61aeecb67db282a75ec5:Lithoxyl6
```

Figure 19:MD4 Hashes 4

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

```
./hashcat-cli32.bin –m 900 –remove myhashes-9-raw-md4_new.txt
dictionary/wiktionary.txt –r rules/best64.rule –o resultsbest64_passwddb.txt

1.  7f042892eff29b3605074185033ee593:Asiamericana
2.  618ec00363a4a97c091be85a140fd5e6:Anasazisaurus
3.  50f1344f667719663931afccc76c6133:Atanatis
4.  a7a6ca6c68f7184f7c973c87487ff2c3:Altirhinus
5.  b0c6546081e90eda99aaafa606fd4336:affublas3
6.  c065319c0a506cdf92cb523da3a43e5e:Bassiliou
7.  42004fd3a6e3200f086fcf78b0c1d013:Bodykins5
8.  017188c9477ab5436329366fd38ee01f:Compends4
9.  ad3a19b0f7ff09d70db804a09e3c26b2:Camptonotus
10. b00f8285935dbcf89bc7bb24702e4008:RETCENNOC
11. a75777ab7eaeb04ed761c1848c030491:dancecode
12. efc301e56913f5fc019f592795a7fe00:Chronopolo
13. aff5a83ce6a8bf2218f32fe964314c7a:Dichtstuk9
14. 14e3527ff0045c1474aca8e647eebd57:EXOCRINE4
15. f4a002586b60ab8c4354cb082af91ac8:Forgifer
```

Figure 20:MD4 Hashes 5

Summary of the attack on MD4 Hashes

Wordlist name	Rule	Result
Passwords.txt	Best64	158
Output.txt	Best64	91
wikipedia-wordlist-sraveau-20090325	Best64	43
Passwords.txt	Specific	23
-do-	Generated	146
-do-	Combinatory	6
Grand Total	**--**	**467**

Figure 21:Summary on MD4 Hashes

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

4.1.2 MD5 Hashes

Hash Code = MD5
./hashcat-cli32.bin –m 0 –remove myhashes-9.raw-md5.txt dictionary/passwords.txt –r rules/best64.rule –o best64pwddb.txt
1. 98bd1c45684cf587ac2347a92dd7bb51:last 2. 5fc25157650d0cb24f02216d904584df:plan 3. 5547a0858b01680be8f64406a3050b8d:icky 4. c1d9f50f86825a1a2302ec2449c17196:H 5. 6c92285fa6d3e827b198d120ea3ac674:here 6. 0d149b90e7394297301c90191ae775f0:it 7. 13db2d9fb60836915205ff15c64af1d9:Again 8. f97c5d29941bfb1b2fdab0874906ab82:one 9. f07849034582c8c26466a13387a6c1f5:5AqrFKf236 10. 0d1a6020a240b35b80df111bb6fb9ece:<u-pRinCE 11. affa5249363911dd655e5a7cfacb49e6:Dora 12. 8f22e8ff6e2be10b6bc7e6cf7b47d782:Miller 13. 622f90ceba19c667b7d2620169144476:afterwards 14. 0fa3f2feaf0b81746a28fb9779a6cf30:Dais 15. 823c20c7daee65d9928c3cc82767d827:Agnost

Figure 22:MD5 Hashes

Hash Code = MD5
./hashcat-cli32.bin –m 0 –remove myhashes-9.raw-md5.txt dictionary/passwords.txt –r rules/T0XlC.rule –o T0xlcout.txt
1. d7a5b8d1421a14a96f916236ab5f2736:!@#$%^&*()09 2. 4c80bfd09d9795836459517c51c585bd:alameda78 3. 1a966c1a3cd461db6e7bab047ea090fe:anselmino 4. c20109ead19b0b9a44103c96eb10617a:alexandra!65 5. e9fed54d61e81b99d4499b92b6da3c3b:Bbellinzona 6. 3c7e34d31a3ceb611e42f60d8de061a8:onburrell 7. 4a40ffd5df3f3daf4c8940992a5feb16:a Student 8. b807001856a0847eb899e69db02eb7ef:Riviera# 9. 1ea398c5747bccd4716105d7381d8f5c:Roping666 10. 8ca14d5270735adb8de1da3988fd99b9:merrill45 11. af916e35b39f757bdc86348c66a4cca5:MISHA7777 12. 9f7fe97f5e8d14904e070c06f0706a7c:moimoi861 13. ad12f19003436b18e66d902659de9861:the Trask 14. 91f48cf5ef2d8a905fd0dfb7d01ca9dc:PAKISTAN25 15. 707546f54f8490c15e2df30ce86e93ed:Lanayiota

Figure 23:MD5 Hashes 2

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Hash Code = MD5

```
./hashcat-cli32.bin –m 0 –remove myhashes-9.raw-md5.txt
dictionary/passwords.txt –r rules/specific.rule –o specificrulepwddb.txt
```

```
1.  6a7aed12afe3aea477ce3249db70919f:1158vette
2.  880749269207e3e2bb640fbf1e3ef25b:0177253464
3.  083b2951fdea185b2df0680d802e50f5:pAassword
4.  4ba20899cc0e3e3a8c98798ed3786fbd:April1675
5.  0a1eccfe7d211140d25b10389de11698:068419959
6.  35f489e4d17451b435215adb75c9e159:360914269
7.  1cab5e4a285df9b8b91ecdaf013aedca:Scituate.
8.  aefd75fa9d136c19e52e79459a559c8c:Dioetikon
9.  3d576a394159a4f0ee4e993e064ad104:Bobrinsky
10. c76b3272fe7e843afe312d92da430896:cartman64
11. dcc8d382471a712ccacae0d35c6fcc9c:aisystems
12. 5e92d8a5ba803b6f443987620ca15177:chamber56
13. baf813a13247dc422fe06fdf16837a3f:ciaporras
14. 1ec811e8bfb7e19ebf74e9f377e025f4:mfletcher
15. 2efc0c96e72068cb6b28cd9fe15fe9f7:gujerathi
```

Figure 24:MD5 Hashes 3

Hash Code = MD5

```
./hashcat-cli32.bin –m 0 –remove myhashes-9.raw-md5.txt
dictionary/passwords.txt –r rules/oscommerce.rule –o oscomm.txt
```

```
1.  bbfc489d74c32b56059f7c5a69e3284c:467victor
2.  332468dea23ef95ceaa88a72f68ba783:6782830747
3.  c61aad23b7f6f78a45d6e199ce312966:9115151782
4.  0dd5bd1f0fd1ee1f5be20856b2f22498:12Tacitus
5.  df4389b0da451c417d5ddbaf26a5f8fc:18finance
6.  35972945b3d81d804084471062ec2e5e:93macgyver
7.  cd66934b849435413e273cfcc1ed7654:67malraux
8.  da00e1955ef6fa25b04ff007e219b191:46lysiane
9.  131bdef049b3c7fa340cec30cfb3bd26:12prenses
```

Figure 25:MD5 Hashes 4

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Hash Code = MD5

```
./hashcat-cli32.bin –m 0 –remove myhashes-9.raw-md5.txt
dictionary/output.txt –r rules/best64.rule –o best64outputdbmd5out.txt
```

```
1.  59f4f1ed6ed9d86ec215dfb6be94b524:923890!@#*()
2.  0dd5bd1f0fd1ee1f5be20856b2f22498:12Tacitus
3.  131bdef049b3c7fa340cec30cfb3bd26:12prenses
4.  17799db4610180e9017222bf8400758a:097870911
5.  00031fae7e7626f49ccda64fecb5bf3e:230272988
6.  d5a5a33588c7bc9740cef46f95c3ffc3:200877477
7.  5b95cbbc330b39efe072222f977868bf:9991masha
8.  ebc58c4e91f3c13d5de90d4e0cfc60e3:264461777
9.  11b62117c3e8358715c97bc27629e787:208566pao
10. 8066bbefe2f38b27496c18181b353340:1TERESITA
11. ab0896974dfd0e69f831ceb1d4318472:Arretium2
12. 27b1b402583bcb31515bdf11117ebf60:ERIKA1809
13. 5d639c3561d716b6e884ab07d24e7250:Conomos
14. bc60ae30257c3e357b0a6cce1ff139d5:BOCCASINI
15. 1ab8cae3aa5518edb3f6a6915d46bc36:mEHETABEL
```

Figure 26:MD5 Hashes 5

Hash Code = MD5

```
./hashcat-cli32.bin –m 0 –remove myhashes-9.raw-md5.txt
dictionary/output.txt –r rules/T0XlC.rule –o toxlcoutputdbmd5out.txt
```

```
1.  3aed19370ae89c38e06351eb18efef59:050902507
2.  fc34bf2f28f15fc94e0b02f907bad433:148640529
3.  70219357e150122a60e8f19d69d598fb:089849507
4.  3bd3e46bbfdd769f193be27ec6be1e4c:034499030
5.  1922ed54fc63c76277fcc194b058c5c0:034531166
6.  065317a52a6cbe3b71749281c2430f1f:132353166
7.  dc7dd1df39135f520b567d7b4167111e:097654796
8.  38a42d03e197bf75aa67d174918cd64a:013205970
9.  b5653aa16338ba5608f10dd4ee28974d:138026842
10. 2bd91ca7a2f7a938e6cb798b5386f762:045389408
11. f7823c72654f87149b4b223792b114c1:138711596
12. 5fe7b6cacdfa874e9cca675ff57913dc:120516704
13. 2f9dc190c2b21a83bac5b3db86729b56:0182633674
14. c72fdc65ee61aabd4f61f0cf58a4b4c0:019270934
15. 49bf275ccf02b33fce5ee6899c927d04:293955110
```

Figure 27:MD5 Hashes 6

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Hash Code = MD5

```
./hashcat-cli32.bin –m 0 –remove myhashes-9.raw-md5.txt
dictionary/Wikipedia-wordlist.dic –r rules/best64.rule –o
md5wikidbbest64out.txt
```

```
1.  ca5a6a99dfaca8d48e6a43d7180bd701:anaghabha
2.  9b66acab68cc641cc431aef69a516ec5:biscopdom
3.  0a4a1db80b5517448667a60131f1ca98:Dinotyrannus
4.  6c4777b127ecf0e2b95175f7e2e0f53d:directans
5.  6b3b794e31705f958e79eca962c23c50:Geranosaurus
6.  afacbf6413d2455c23b718fa403813f4:pasterova
7.  a5fedede74d44bad37e43eeda9f13def:Mitsopulos
8.  cab8ef73e323f8dfc46b2aa4931ce660:kiyuyu123
9.  d11ef38c9bd59d4969de381f6f094254:speciys
10. 4a9e5bd1ee5179565b5b57b4c4365d90:Stergiadis
11. 307ea591fb11e9afe39806d34d637f66:Theofylaktos
```

Figure 28:MD5 Hashes 7

Hash Code = MD5

```
./hashcat-cli32.bin –m 0 –remove myhashes-9.raw-md5.txt
dictionary/naxxatoe-dict-total-new-unsorted.txt –r rules/best64.rule –o
md5naxxatoedbbest64out.txt
```

```
1.  adef31231246f3f8cda0a30dac3526a4:on the subject of
2.  6f3410a38f235e9da4c1338d1203fdda:pass the time
3.  3b9e2db8ae741a8d1dde6bdc4713c43b:Fr@ng1ble
4.  bbd15f16362564066eda65d3f2c05edf:Pu$$yca+$
5.  8c6c772882146f9efd416a141a0923c7:Respec+ed
6.  4be79e3b3506edcc9b6761f6e47b492d:18-9-1994
7.  7b68201a65bb5f952e637a251685f636:tracymets
```

Figure 29:MD5 Hashes 8

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Summary of the attack on MD5 Hashes

Wordlist name	Rule	Result
Passwords.txt	Best64	193
Output.txt	Best64	57
wikipedia-wordlist-sraveau-20090325	Best64	11
Naxxatoe-dict-unsorted	Best64	7
Passwords.txt	Specific	21
-do-	Toxlc	91
Output.txt	-do-	45
-do-	Oscommerce	9
Grand Total	**--**	**434**

Figure 30:Summary on MD5 Hashes

4.1.3 MySql-Sha1

Hash Code = mysql-sha1

```
./hashcat-cli32.bin –m 300 –remove myhashes-16.mysql-sha1.txt dictionary/passwords.txt –r rules/best64.rule –o best64mysql-sha1-pwddbout.txt
```

```
1.  594c7d3a8c9e120105bedc7b6e41a74ab96533c2:Rufus
2.  a6e43ad4dab4cbd96d96e7087bfc07fabfd714ab:back
3.  ae1b02b5ea66ff946b79e87b5d80cbec30c88b3c:rest
4.  a598af17107ae6c73080306b92c81eab0badff72:7Z4iz0c379
5.  6aac3f4e790d3b9303b169a460a9d1daa5427c57:Fred
6.  7b2c813c00a0eadd4709e10ee144934d8d3bafd7:sunset
7.  3abc5018ecb6bb501ba50ed6fcfb3e15fa35396c:true
8.  1b2c617f0a5739ea5f7b152d23ad984b572393a2:Hannah
9.  ef3fe3515901c4f1f79be535e6b5d95ea7769ed0:Fram
10. 9c0388734aff1180c7bf56df8ffcf6d16cfcbc2f:Mimi
11. eadaa2161cc3e7949fc003a22bd0cb7899635299:403320106
```

Figure 31:MySql-Sha1 Hashes

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Hash Code = mysql-sha1

```
./hashcat-cli32.bin –m 300 –remove myhashes-16.mysql-sha1.txt
dictionary/passwords.txt –r rules/specific.rule –o specificruleout.txt
```

```
1.  44ef87ba0709efd11809a5b9ed6cfd1d81b25dd7:!4potencias!
2.  40f5d847f5f36191622824e1f635ae64cb23d103:Agressor24!~
3.  b2397da6520b90604fa51f346c37960faaa7b04c:1986323215
4.  21dee375c16c243b1e47c0c16168a6350aaf8628:29sep1967
5.  fcb16333bb2ddc91211f7a8661d0ca4ea79e682d:Anderson>
6.  f84fd617de4b2a375b5a49c1922c836653f9d2cf:02478547tt
7.  4cb28b0503258d21c911c5a9a4b77fd42e8617c4:1300220359
8.  1cf8b6749e7549a1ab04cc9e64201f1dc3f26d11:431884130
9.  b207722736239cd172eff003308095f055ed355c:1v1c@k3k3r1c
10. 80bd42c92c1e639b6ba33cb9e19887c62667d27d:0011Getinfo!
11. 083e2c5378b4e82f2e2e31de35307e719c9c18ee:Sumnos
12. b24fd8686131c3001f9c00ee827c0662151ab528:Bridget44
13. a5903a7f8ab00155534e72aad11032beaf0b47a7:Ellen1902
14. b9064c0621d2027fb34246a871b9f97c6990adb1:Cayias
15. c775e90639c55f176beb26dc09b79db88018d509:cristacar
```

Figure 32:MySql-Sha1 Hashes 2

Hash Code = mysql-sha1

```
./hashcat-cli32.bin –m 300 –remove myhashes-16.mysql-sha1.txt
dictionary/passwords.txt –r rules/generated.rule –o generatedruleout.txt
```

```
1.  4bf91a12e0275652f01d9fbc5caa4172fe64d625:927celine
2.  ebabcabeacec6efb78116a6a7fc541558505b693:oltzinger
3.  d06755335f28a7d2b0ba18f68803317949bc27b0:1182toine
4.  2bcf036e326f29980b8d390c1a76c5b604862487:Capparatus
5.  dad521586af60c67fd0f4b5de4cb74c6c2a85c45:666666mir
6.  12f1668713450424e43b83dc787146f67533c515:125016516
```

Figure 33:MySql-Sha1 Hashes 3

Hash Code = mysql-sha1

```
./hashcat-cli32.bin –m 300 –remove myhashes-16.mysql-sha1.txt
dictionary/passwords.txt –r rules/oscommerce.rule –o oscommout.txt
```

```
1.  bee1c8161119f93dcfabb0e79ea54be0dacf2989:984578430
2.  77324f2e1e1de7478aac62c002a4ee66309c1e70:413004484
3.  32f372d8b398de799960c24d5308f382b4d1a7c1:215550777
4.  f7c86c8050214f3689b66858afaf6b3ee9649adb:417895584
5.  bca568a3e56daaf132a0c91d93a21dd05a7dcb4f:16august06
6.  23ddc762ca7d8b2b3da8c01cab7efe0686e2f99b:acforsale
7.  b8c8e07a7d189bf9219832df8a9ee52de0cf6867:bfrugby11
```

Figure 34:MySql-Sha1 Hashes 4

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Hash Code = mysql-sha1
./hashcat-cli32.bin –m 300 –remove myhashes-16.mysql-sha1.txt dictionary/passwords.txt –r rules/passwordspro.rule –o passwordproout.txt
1. 757d3de683dcc2222884b9d42d11d22d3160d1e6:25FEB1966

Figure 35:MySql-Sha1 Hashes 5

Hash Code = mysql-sha1
./hashcat-cli32.bin –m 300 –remove myhashes-16.mysql-sha1.txt dictionary/passwords.txt –r rules/toggles1.rule –o toggles1ruleout.txt
1. f430c5a32946aeba9a37ff5a6bef5e268e7adcad:lipomyomA
2. 42c176d6c2f00b335b09c4636f4c4eba017a0ee8:papolateR

Figure 36:MySql-Sha1 Hashes 6

Hash Code = mysql-sha1
./hashcat-cli32.bin –m 300 –remove myhashes-16.mysql-sha1.txt dictionary/passwords.txt –r rules/d3ad0ne.rule –o d3adoneout.txt
1. 8ea22439d82caa8a6f87c6f3cc4d253ca1c585ab:passwoSrd
2. 56a9b3c5593c1eba1f47bde4883308529153a1e9:893261593
3. 21d101a8702726e37916d63c59575451b4c21c19:(*(*@#$(2333
4. c558ed06b332c9b285db15f16d986c8fbb2ed67a:1jmandude
5. d98355098eef35429c42b046691be6142ebf2912:8P4ssword
6. f16128538e9cd591439848616b75fb4f6e4e72bd:4gandalf8
7. 983de402ce0f581fbf31050c16c06a366171503f:119espoir
8. 969d58bc51d2fd89adf6478c3319274ea8f15e99:620778412
9. c460743c4718fdd80deef2ffe4437c25fe389637:Athanasakos
10. 77456c422ec18c74422091ee711a541b64e29f5f:AarbergB
11. 388e9df98f92cd528b5a2af8d31d8b4b289750f5:172019987
12. b59f26ce68c4d13b836ae9a59063d42f42f390cf:916787784

Figure 37:MySql-Sha1 Hashes 7

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Hash Code = mysql-sha1

```
./hashcat-cli32.bin –m 300 –remove myhashes-16.mysql-sha1.txt
dictionary/output.txt –r rules/best64.rule –o best64outputdbmysqlsha1.txt
```

```
1.  6c302b953b86c0a9113ce822810f26a5a052ea98:123qwe45f
2.  92e7831d40edfd7282e90bda746343fc64898dcc:086265112
3.  6d49a90c685fb05b9b5674689b9cc99fbaff57f4:111468169
4.  13f879c3a464f493eee47fcc2b24bed361a3a803:132108611
5.  0046a221614c38db76b52a37a9a0c5ab5c4176f1:04.04.1980
6.  87f2b7db3bdc0d035b48830fb30260b633e828e3:229930611
7.  207db8058901e611144cabd3ca1ead91354ed8d7:541884969
8.  23c84b155ccbec84224136f1c5013df7376bd749:586164940
9.  3fe588a9c6fecf57c32cbc6384468c9919efaeda:786439969
10. 7ed2d00a952b2776c0b7452b526e7d54b1e950c1:6c019n3pv
11. 7897c6532882b677e9b92cf87f290e4597b0f067:8907832qq
12. 014ae3a08bade109e8571d166b07afb367bacfe3:Alegrete9
13. 5f5957783ef3981249d0caef2bbe523671327e8f:LNARINARI
14. f89dedb3e524cf213178dc4e3c305e8853810bf1:Mellific2
15. 9dcda9bda7140c44cc14f1d47f24eef4559d85aa:Elosaurus
```

Figure 38:MySql-Sha1 Hashes 8

Hash Code = mysql-sha1

```
./hashcat-cli32.bin –m 300 –remove myhashes-16.mysql-sha1.txt
dictionary/output.txt –r rules/specific.rule –o specificoutdbsha1out
```

```
1.  4faad38b1519d18e2b7252c20fd9f21305447ab0:087941978
2.  cf500603871bbfbf0dcdb8ddacc601ade7db0eb9:069453620
3.  0c99d568cada90cebe2716f0f59f37778b7ff840:/canvasback
4.  df529050588e2f840ba25139214b6fe3531d50bb:1890361239
5.  c5aeb28026d3bbad2e924ddc8ac82ca31c5c0fb2:471502040
6.  77a186aa05a914974c6e9ee5ffa347fd146d0a47:@@@555777@@@
7.  9af458d8e7b9c7a52ceee3736ec209bcebdd98c5:95422993wf
8.  4d7d18a5c88da0dae3cb3b53404196129728eda7:865431234
9.  8578a4b89e7c1f2caf4c4154d75d0d9eeab9d361:Alonjus
10. 09513e1bcd13a45ae6c0cdbe8d3bd2c1b3c79563:717866747
11. 54b011788ee5d09fd6e83dc414368d36af92af6c:708378013
12. be44d2d96c85e3a2a5b4a2863630863d1f57df33:Demdness
13. 6ad6212503608793d7470139facd07317c0afc65:Lulfis
14. 1a8bd347a4d38d77cba7fd87feb294bfd9b08278:a4!@#$%67490
15. 01ffbbd3960ed9907fda3bf61b0769c2acecb6c4:Warcraft67!!
```

Figure 39:MySql-Sha1 Hashes 9

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Hash Code = mysql-sha1

```
./hashcat-cli32.bin –m 300 –remove myhashes-16.mysql-sha1.txt
dictionary/Wikipedia-wordlist.dic –r rules/best64.rule –o
best64wikidbshamysqlout.txt
```

```
1.   97a9c20d972700e615452501580a6ca8fd047c20:Anabisetia
2.   3c52fc7c3cf8f6f7d7bb2d013fe580446864199e:Anastasakis
3.   92caed5768ad3e68642d13b711271c2018fc11c1:AIGALIERS
4.   fdabd55e5b9580abfc7dd873db7ab0887835194d:borriel22
5.   c460743c4718fdd80deef2ffe4437c25fe389637:Athanasakos
6.   973c8a334e24346a60fea58728a971cfff7c260f:hcconcord
7.   a627afa9c674d399a684571532396043b7ed32f6:Gatsopoulos
8.   2e386a864a5e2afe09a8f65f819afe8d1fae9d36:jonreuben
9.   544efb1b779bf0cc3005710ffebdcb9514667be5:hrsonline
10. e8b5ef69c9771fdeebd0f7dcdf6db2e6461b9050:Neosodon
11. 367efa46ecffbc872e6dd959cade016b8ccd2d39:ritastar7
12. ebb1af8ea6356df8b4ffb71c72bd7e928678c6f1:rageasses
13. 1969dd7ec507154db853ef0ed93a5f91d346a3ab:SUFFRAGED
14. f4c18e87822f8603ba3f13059574be35774aaf9a:Sanchusaurus
15. 624858061dde3a3a89b007c1f9dfa9b3f776f8d1:RUISSELAS
```

Figure 40:MySql-Sha1 Hashes 10

Hash Code = mysql-sha1

```
./hashcat-cli32.bin –m 300 –remove myhashes-16.mysql-sha1.txt
dictionary/naxxatoe-dict-total-new-unsorted –r rules/best64.rule –o
sha1sqlnaxxatoedbbest64out.txt
```

```
1.   010ef141e6b781dca8413e16dad226289ce00b2a:FLEDGIER7
2.   94716b26dc67cb21e1c781589ceedc0d8321af51:ENGRAFTS7
3.   db650cf2ebcf8752897d03940c3558a28afc213f:Stereoeds
4.   4cd3d5df9b4f1eb5f9735f6e708e6885c9ba29df:PRINCELY0
5.   bc780c259130c80c75cfa09a0a8d7afae89b0ad1:ROTTENER3
6.   0dbc6ad1105db263fd1c0bd2bd0eb2074e48aaa1:Propjets2
7.   e9f94237b9c372784c99ff0db2267e0acdf688eb:SOURSOPS7
8.   2da2e046c49a965c09719769354d578fac92d7f0:Alg!c!des
9.   64620d5079eb7a0a9c474b7ad57163aba79cb410:exc1$emen
10. 61c5c9e78b82df12834e0e254063db85192da068:HINGELE55
11. ebcc2d99cdc1f6c8a9416eef25002ae0d4aa6426:Cor+isone
12. cfed96112442d1bceef2cbf8c13b6481b9f4c13c:m4tch13ss
13. a96cbaa349eebe239f2cc6472a325e6b29ecb454:STEALIN9S
14. 00399771ba3681a801e3b4b4fc94282adf5a6991:hellguest
```

Figure 41:MySql-Sha1 Hashes 11

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Summary of the attack on mysqlsha1 Hashes

Wordlist name	Rule	Result
Passwords.txt	Best64	245
Output.txt	Best64	104
wikipedia-wordlist-sraveau-20090325	Best64	17
naxxatoe-dict-total-new-unsorted.txt	Best64	14
Passwords.txt	Specific	33
Output.txt	-do-	25
Passwords.txt	Generated	6
-do-	Oscommerce	7
-do-	d3ad0ne	12
-do-	Toggles1	2
-do-	Passwordspro	1
Grand Total	**--**	**482**

Figure 42: Summary on MySql-Sha1 Hashes

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

4.1.4 Nsldap

Hash Code = nsldap

```
./hashcat-cli32.bin -m 101 --remove myhashes-6.nsldap.txt
dictionary/passwords.txt/passwords.txt -r rules/best64.rule -o
best64nsldappasswddbout.txt
```

```
1.  {SHA}/dkfusDlZUpwAmoinMil2GkVeyQ=:trip
2.  {SHA}5+aUxYzVDgMk7JaRiAC8Nc0XYps=:book
3.  {SHA}GCC5QnLd/sgD0vEXid3plVkyX40=:cure
4.  {SHA}LdK+1yVni0+WPyfV6Eek8h2zHNE=:today
5.  {SHA}QJrnidrWdaYfjIcFrnU6Cm8au8E=:Emma
6.  {SHA}aj+zW2dLTDoZeNoqSVtBHrovw8A=:Laura
7.  {SHA}tOPoHGUf5JSt7DZvlLrwV6lN9v4=:morrow
8.  {SHA}3nmQ4C54AaWGeS5NI8WNbiED4Fc=:7fX0Bz9591
9.  {SHA}TD46uO4cGGAecWHXgQ6ip0ofHew=:Honey
10. {SHA}XlzUsFllmFr5x+Z+wu1R0H5gRrg=:5Sji7qlp7X
11. {SHA}JBiaaBzS3MmebbEM0ghiAXWqdoI=:+garecaan
12. {SHA}dyD1h94qgB81tXeGrJCqLQRY7tU=:24295865ab
13. {SHA}D1jVpVFfGoqdF5qliFi2ey+KM4g=:000000000
14. {SHA}943JJrAYP6HKT07c1rHmu03i/ZM=:3fcqtD2184
15. {SHA}zAQikkdP21iXFCJY5pe6D4uUEVs=:Andrea
```

Figure 43:Nsldap Hashes

Hash Code = nsldap

```
./hashcat-cli32.bin -m 101 --remove myhashes-6.nsldap.txt
dictionary/passwords.txt/passwords.txt -r rules/toxlc.rule -o
toxlcnsldappassdbout.txt
```

```
1.  {SHA}RJVWOgxDsvTSaQISaXZJd5sRMR4=:10161988S
2.  {SHA}ZOcjq4SAVkDAdU0YeY0fcEjaEH4=:117190756
3.  {SHA}3jWN+V3fZkk4O/5oIdjkyoIAT4U=:Aarber|g
4.  {SHA}qtur5eBhIzh9oOU9ZfM5qvPKils=:abdulah36
5.  {SHA}YNpVaz8T5tHPpLdk1ZpKxjCmL+w=:Alcyones.
6.  {SHA}nrzUomLAiyYwAs3GlSuz+jQ0MMA=:297232147
7.  {SHA}N1XSzo83G+ikYbCUqCcRDvyf9jw=:xheitmann
8.  {SHA}g2rMfzFqpJW+tFVOo8Bc02uLP9U=:Appenzelll
9.  {SHA}QSJSUdSVMlOJkHv75wlBFHTJueI=:49th@mail.ru
10. {SHA}SFnIMk98q1AxpZQJe5L73QZHEmA=:131294thi
11. {SHA}Xu5BgTNtvsqp5+hDDXtRSxzJxbc=:Ausgeheckt.
12. {SHA}zQ/P6eVv72TlfUljjL0CtbYweFw=:Austin1736
13. {SHA}7uuJLGWqtFHxjykEQPyaafu2zHw=:1987axime
14. {SHA}VhTwTmnLRk6pXp1JAIBLpeNA1CY=:501435572
15. {SHA}zsz8uekzvGI5UO9IyrYaRTS97lU=:afterbits
```

Figure 44:Nsldap Hashes 2

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Hash Code = nsldap

```
./hashcat-cli32.bin -m 101 --remove myhashes-6.nsldap.txt
dictionary/passwords.txt/passwords.txt -r rules/generated.rule -o
generatednsldappassdbout.txt
```

```
1.  {SHA}+CwFFoIFcrM2mRAdO1IO7AY/0bM=:!@#122QWEqwe
2.  {SHA}CzGevwrNWv0Gt5N05GTgvOa6oxw=:870420156
3.  {SHA}Elamn4dWEMI+OKuI/ofXe0jrQdo=:954791631
4.  {SHA}9YanojYMk1OUJt5Kya6xthPak0M=:$pastille$42
5.  {SHA}Pl+icqkzmXl0ymKU4TwmJUy8Ooc=:203azerty
6.  {SHA}RZ/WzhTHb/SClwbCMBxwIApCQzQ=:13jun1999
7.  {SHA}guDx2L9aIJep+0xnCP/FLGDFQ7w=:9rVf5LD561
8.  {SHA}mkvG1RwVJOuC8owyL/8Lzl/zUCg=:10Jul2000
9.  {SHA}8u0EToI3FenNnWq7U2IQhnCsXds=:@ll@lone4527
10. {SHA}vYTrl3/r7+BNNH+yLVx0Wk35Yho=:4April1010
11. {SHA}ncgnIMpWZBtHG2jsT5DafE/tcuw=:195741822
12. {SHA}1dhEDCDhcU8K/tMONs+0eYvZcAU=:AArdvark33!!
13. {SHA}Ld9Bd/gjEtoJzY+lSbawAURnvqY=:nnnnnn494
14. {SHA}DV6KbXYIxoX7dQ6vE6dvhFClCsM=:COXCOXCOX
15. {SHA}v1PnLsaL9t04cBzCo+8gCuO/FAI=:ACCESS1900
```

Figure 45:Nsldap Hashes 3

Hash Code = nsldap

```
./hashcat-cli32.bin -m 101 --remove myhashes-6.nsldap.txt
dictionary/output.txt -r rules/best64.rule -o best64outdbnlsdapout
```

```
1.
2.  {SHA}Q1NyWd4sVqLUmPTpjWKSPiAGPK8=:054340121
3.  {SHA}2DZ3Of8ylJ/XWALU9+1jGrY/G4Y=:083588899
4.  {SHA}nGnJkhue4NwUH1tADyD0ylormAA=:127608799
5.  {SHA}nZyEYerkqU+IboPFSXd5A6c+j60=:12NEIMM12
6.  {SHA}M+F3YXRT57vhJ3HiDJuZ8KAnTwk=:00rakotas
7.  {SHA}iK9ddOGF2HZ8DSilFKJz3RLufgw=:96.11.1989
8.  {SHA}xmj00rMTDXrCd2qfTcm0agQn+cA=:185861800
9.  {SHA}70DelroMggpqjbiaV2rJvu5BoBM=:2t5p0thlz
10. {SHA}thZOX7wPQvYisx/a5jE1pYB7Mfo=:653450960
11. {SHA}eWPoIsoFMb88Hze1JbIOi/oO9xM=:48135svue
12. {SHA}0GXXPxOHxxCYA8uK48xQcFIH2JA=:666luc999
13. {SHA}QmFN/p5Gkd31el9vhumJsU4e+3w=:AFRICA3A@
14. {SHA}JWIusSONYdH8rRKtt+u8Fegc1nc=:597291733
15. {SHA}vuozmhfjWSPB20FGS6fseWiiQpM=:51%fck07%
```

Figure 46:Nsldap Hashes 4

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Hash Code = nsldap
./hashcat-cli32.bin -m 101 --remove myhashes-6.nsldap.txt dictionary/Wikipedia-wordlist.dic -r rules/best64.rule -o best64wikidbnsldapout
1. {SHA}DrmUiv6bBJb4Zcun0i/XwPYbX+c=:Brohisaurus 2. {SHA}XImZDVwurlg/puRkxDVRXBYSamA=:aratamaya 3. {SHA}l8Vxd3ASnqZkzpyIUxWUIISOcTA=:Bryennios 4. {SHA}LpaihTPpi4VhcrykP1Vf05zKHX0=:Dhikeos 5. {SHA}uJRuA97Fcs3QukFAGq2t5YGhUys=:janatpour 6. {SHA}WqBO2qLqKJZwosyWD8oKhhXqOEA=:Gripposaurus 7. {SHA}ZgU2OaZGagXqyvOEkBf2bZmHOKI=:hugels123 8. {SHA}83k3n+utZLQM+IeIQ+G6Q/AatIQ=:Polykhronis 9. {SHA}S0GkReetW+B1pb5y3Fhzxu1yYSU=:Salimosaurus 10. {SHA}B/tuZS7UEk/PY5XaLymuvUYwqno=:Valdoraptor 11. {SHA}nNz6/Gc8YN1B1sFeggK/tzA2eoI=:Vozikis

Figure 47:Nsldap Hashes 5

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Hash Code = nsldap
./hashcat-cli32.bin -m 101 --remove myhashes-6.nsldap.txt dictionary/naxxatoe-dict-total-new-unsorted.txt -r rules/best64.rule -o best64naxxatoensldapout.txt
1. {SHA}JP5L7AlZ/lZsYy3qliwTpyo3Ve0=:Catherine of Alexandria 2. {SHA}p6E6W5E1wld2IGAdF173tW64WhE=:bab00n1sh 3. {SHA}1ie0IJXnzIEFIL2Lr792RnDXxSw=:B1az0ning 4. {SHA}bMfGE5U4SFvs47CS5KxgJVy7BN0=:chamoi$3d 5. {SHA}vHJ+2RJiREK1xV5TXa/3KYRxEFo=:f1r38all$ 6. {SHA}HAJsD4jQRm6207oCxRvVv8x+Rsg=:OR9ANIZES 7. {SHA}nU2LEoHkpiLa1wu7whVJg+SkEj8=:R!gm@role 8. {SHA}erQ5WkANg10PlBjlZ8hyruI9eKo=:moon83ams 9. {SHA}wxs9i1uGWdthusCjTvouuq3HILo=:Skywr!7er 10. {SHA}Z9zpQSREm2jVWdKDFpgc3+ouz6c=:vickyedge 11. {SHA}NrONTfupnuM9Jp3SjmMoknCUAM0=:JEANVICKY

Figure 48:Nsldap Hashes 6

Hash Code = nsldap

```
./hashcat-cli32.bin -m 101 --remove myhashes-6.nsldap.txt
dictionary/passwords.txt/passwords.txt -r rules/d3ad0ne.rule -o
d3adonepwdnlsdapout.txt
```

```
1.  {SHA}SYJK2AW/yxwSdwi1IqACEDkpnrE=:038481598
2.  {SHA}yi2bNaDUwzidMt9628NSf3gh3DQ=:@hotmail1994
3.  {SHA}/mm+MoH/W1a7AGSXywwGWAGC1WU=:@zezette2009
4.  {SHA}w8CqtPTKc7eBmGFzv4wXIoir6BY=:248712435
5.  {SHA}dqIFDRA6VEcO6R5vxA+liOJNgoQ=:0121410011
6.  {SHA}Q/lm8USZe5IrHGGRzjicJhaeoZI=:362332688
7.  {SHA}SO0eArLkigPdz9KS2xqmTpTc3qw=:Aarwa]ngen
8.  {SHA}jY0OJUZq62YRyUTwiBshQ5LBJoQ=:1219MARIO
9.  {SHA}B0DNIQyucrkZi051ArQjeCWzKkk=:777929455
10. {SHA}qDZD6Rfua1sYbnscc0g4Ro5zvq8=:470523820
11. {SHA}Si3qfBDarTdZyAYi5nL1oejk/p0=:Anologie9
12. {SHA}wDwxgZBnF0pVL7DII4JCrsloX+8=:455263588
13. {SHA}bfwQwVIyNWUMWbY+Up17uDq8R1A=:906223008
14. {SHA}SKpIoR82FNYk2bco4ujNejmq3lc=:114402829
15. {SHA}tjYC0Z2F4JaYb/H+cM1xROZyJUU=:700915972
```

Figure 49:Nsldap Hashes 7

Summary of the attack on nsldap Hashes

Wordlist name	Rule	Result
Passwords.txt	Best64	327
Output.txt	Best64	101
wikipedia-wordlist-sraveau-20090325	Best64	11
naxxatoe-dict-total-new-unsorted.txt	Best64	11
Passwords.txt	Specific	1
-do-	Toxlc	182
-do-	Generated	325
-do-	d3ad0ne	29
Grand Total	--	**987**

Figure 50:Summary On Nsldap Hashes

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

4.1.5 SHA1

Hash Code = SHA1

```
./hashcat-cli32.bin -m 100 --remove hashes-11.raw-sha1.txt
dictionary/passwords.txt/passwords.txt -r rules/best64.rule -o
best64sha1out.txt
```

```
1.  19f319efbd6da4987012e838ee7d7f95e6f70911:shoes
2.  3cbcd90adc4b192a87a625850b7f231caddf0eb3:word
3.  863cc374e6dc98095eb06aa4025895b4a28df6c1:thing
4.  bf93e5ce8bc1228c2585b3f5a368053c9fe8346a:Down
5.  7e9219a0599eae1d9601883f894b4fbe60870586:can
6.  98a91498122b69a070229dc4bc758e5e6633a90b:Denise
7.  a4931f9563a8f4943b4b94af45ed7073d77aa8a6:hipp
8.  e3b82040565bb4be6b11f778a2e3df327ed20a3b:Now
9.  44c8361555e5a6308bb523a5faca6852b0f1fa99:purity
10. 11623fa2a82aba3964f9b9697b414cafaef81b8f:Alexis
11. cdecabe1520a398e4e46deff9d55b19bed6d6457:escape
12. 796ea986ecbaf876d3b3667dd2795656d42de1d7:5k47HdwiyS
13. f8a902b067d4344b687920f92c669bffad7b0e0e:TB
14. dbcf6d6ba1c030decacef981a87a1ccc1e27fe38:Milton
15. cf4246301d762e89cb189af00812aa53bb6d9a44:jolly
```

Figure 51:Sha-1 Hashes

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Hash Code = SHA1

```
./hashcat-cli32.bin -m 100 --remove hashes-11.raw-sha1.txt
dictionary/passwords.txt/passwords.txt -r rules/combinator.rule -o
bestcombinatorpwddbsha1out.txt
```

```
1.  b435c089ac2049f07714569a66f18f54de70b9d8:!74Odessa
2.  7118031ccb48ed83f8647ffeb149f83ba405eec6:africa772
3.  287d5dd79690ff276c3f258c55a9c1dc64d046a5:means
4.  e6984a445badd5c6cb0387bb1a07f56e063872b0:tsahi2007
5.  c7020181da61874c9785a20dd20abb1075730031:last
6.  e309b59ea72711b5d18379d459f15f8a3c3b725a:linea2009
7.  77f8625ddfd0571750132d6488dc1f0c8e61def4:food-land
8.  45dcd0ff3272192ec168ddcc218649c9c5bcb9ca:Goulu
9.  c92c5e120f4c24e6f9d2d71c1cf762f37e3e2512:for free
10. d26546865182fce4d67e56509183b0e6a50b20d7:the walker
11. 70039cc797fe0cd2a45aaa4828241ffe7dc9e0a6:they have
```

Figure 52:Sha-1 Hashes 2

Hash Code = SHA1

```
./hashcat-cli32.bin -m 100 --remove hashes-11.raw-sha1.txt
dictionary/passwords.txt/passwords.txt -r rules/specific.rule -o
bestspecificpwddbsha1.txt
```

```
1.  216ae3fc9f9124e748fd4a3c09091fa2b415aac6:p!assword
2.  92290dc8005877bddba256d4d75ce966c3be8969:5clloydboy
3.  614b5927d46c962d21be24c56f57aec7c67030a3:1687@edmondo
4.  f2bd722a3c4b2f552b4c2e128b8445dd2b9c6106:1205d@ve1217
5.  f1e4ef7203c54d9aa36f48533b9cf5814a510929:0127043935
6.  bdf880b34bdce578dbc5f52a8efc068d835cf5cd:125456715
7.  b3a8983d229ed744d2a0716839bcebfdec171f97:123458@admin
8.  f1b222c226c2a691b650dc29f485b6a70fa84bb6:Aliais
9.  e8f08ef4547aef6ebfd81a2f6aebfa93e31947ee:1v1c@k0k1r1c
10. 60330b526915de2d6d7a9e96496e9b76bcd42c95:Aristios
11. e638963a864895e19cc918bb4774ebb676203da8:72patriots
12. 4f513c649e3d14e41547c7438a0b6b2a20010b04:06071987r
13. c6933d9a21ba2203135387d6b69245cd453bf146:748001greco@
14. 92ced4bae7d7e735c1129e0129c9e74b790f6733:562@zhonghua
15. c8bd81d997c397f603ac8c4b3edf3d50224f9171:Lucillo
```

Figure 53:Sha-1 Hashes 3

Hash Code = SHA1

```
./hashcat-cli32.bin -m 100 --remove hashes-11.raw-sha1.txt
dictionary/passwords.txt/passwords.txt -r rules/generated.rule -o
bestgeneratedpwddbsha1.txt
```

```
1.  232944a672d58b5da188ec2b151d82615f3b2b76:19JAN1874
2.  a020127adb0022d87d7df1e4998bea4ff000ffa8:314alicia
3.  809246de563a6622aab863d5a9568665d26ae3b7:731072717
4.  bb3df662db0c1d0f22bb192685566bd37ecb2397:@Teste12349@
5.  ec8a226dc1e3f99ea5f3169cc659d43470b19453:088034603
6.  cc1bda3d153c27b421087c8afc858d8f88be1160:Reparation]
7.  4f67647ff152d191a8f76437bb2711bc26b3d34b:!undertaker
8.  fb7a69cecb5a69e68f21585a9cc81768b7ce9309:11234567891!
9.  85acd1209d1355c247f1adee40f6105d872cbae9:108191132
10. 25cccf49e262a03560f200ae744785e7ac0cf1ee:divorce351
11. 9271299136e1d720c9dc6e8f22fb7c91064490e3:830753335
12. 8cbe8ef96afd5ad0ec91b736d1c130e911dddcf3:4175:4175
13. 65639478d58306aae11cb2cd547727196bdb3905:190752900
14. a184814c8d538b23d08a75e9f1ecb5653ba85405:Aaqrwangen
15. 77fe6c50f9e022ef2cb5d296c30359598ee3ab68:264608510
```

Figure 54:Sha-1 Hashes 4

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Hash Code = SHA1

```
./hashcat-cli32.bin -m 100 --remove hashes-11.raw-sha1.txt
dictionary/passwords.txt/passwords.txt -r rules/ T0XlC.rule -o
bestT0xlcpwddbsha1.txt
```

```
1.  cd23bfdf76293e74d24b9e18104e2efd1e006c46:$uperman9995
2.  885c24c2faa155c39d72590dabdc4e6bc5604eb6:@lpha0mega14
3.  748694b4863ec418bb6c11398c780ea5b31d1dee:c5raphael
4.  8ab0938ba6ce8f08e630bd7a633f0b7a0a95ef7b:TAarwangen
5.  38a1499ef744a9f556949a207397f5e632ab6283:199018859
6.  057a224e9e5167154f2d0a459ffc35002fc6a242:2000199710
7.  1a08069ab206d07e44477a213e6176d57e4ca321:431233971
8.  d4c9a260f8a6474e6fd54ae6de2461eedd3feb7b:659956123
9.  6c31cac58f57a73d162876f7c97d73f5b2c2c505:Allalouf
10. 0b30f0775d21e90e84c1e4b00c56f167cbe837ab:Angelousis
11. 4433fde0b763c4ef3201e8bee61e0867bcb07cc2:123989385
12. 1f847bda7fd10a1ebd554c4b50def33826d78cc4:attack767
13. c3399d1592206ae9286e71d11974d7f24ac78364:AAubonne
14. ff5323031322059c24f775c6b4c9be5e68e79136:321981113
15. f3d48d0a394e0b19eb2b66f06662ffb0807ed01e:511030070
```

Figure 55:Sha-1 Hashes 5

Hash Code = SHA1

```
./hashcat-cli32.bin -m 100 --remove hashes-11.raw-sha1.txt
dictionary/Wikipedia-wordlist.dic -r rules/best64.rule -o
best64sha1wikipedianewout.txt
```

```
1.  eb44dcae41b7fa90c5f152da6a1ea32e99ba4177:armelijk7
2.  7818354225bd5db320887f7da155f600ebf96c32:benagues5
3.  f7ac705d87f5498260a8beef13156f84a6137d00:abdicati0
4.  592b2a85ca3ce5e9892a5af99bdffd9cd2b98fc4:baignera2
5.  d55bd7fb3564c14240b19dbec829f1885435a73d:balayions4
6.  769e9db42aec223b2e180a742fe850c98ac204e7:Campylodon
7.  22f2034211cad781933332e94602be534d2a1bce:CAPAROSSO
8.  e86fc8c7e03de035bbf8d1290edb92781603ee25:additammo
9.  dca6cfed11197a228124899eaa0a5bbe85d2a68b:cariasse2
10. 07be4cef430a6bd50607e4c747f3e180a701d14d:AFFIRMIEZ
11. 490f17e1547ed2e60f363c803ca1043ab5cd5a52:amarrame0
12. c3d8092b45d3d9d30e921c970c341643bbe9cddd:blindiez5
13. d79a5648d7ab02dade4d886a206403dacccd7502:Colidirem
14. bd50a01b979899f6a635910fe7fe8b28999fc04b:Diacitron
15. 7202b27aac0cdf9f2363237d6dbc6ba40b74c668:ermejache
```

Figure 56:Sha-1 Hashes 6

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Hash Code = SHA1

```
./hashcat-cli32.bin -m 100 --remove hashes-11.raw-sha1.txt
dictionary/passwords.txt/Wikipedia-wordlist.dic -r rules/specific.rule -o
bestspecificsha1wikipedianewout.txt
```

```
1.  cced3dd73271c361ea1b56ffe7bf2d510c1d0a6f:ammusisci
2.  d8bfdc1f30258f64d8c8f061e960638e9d335ffd:ecpcentre
3.  173ee062f003936d7c4a35c4eca8d0db983cfcb1:eremeteri
4.  88b539de98156a2d44cc107690697f5c1df6a9ea:dylandube
5.  d89c083c4c6d5dbe4cb5cfd3ecbfef0d8d05d35c:frenitick
6.  f641ff7d8de76b41ca9fe7f5f4b5e4c115acb0a5:macburton
7.  6c38c3b17a6cb25c264455ce6988509f54151a27:mackeevir
8.  fe6e07e9dd0d18a0bd7508ed45b918d7ac7ce501:odspavanu
9.  9aaada869600b71a74b6cd65dcaa5eb049bfc80d:stevecoil
10. 666938a1aa4c18dc4ba272af322446e33a32663e:skrawbury
11. 6898421e49d8d0d622ec1e9499e3407a585bab2c:vinitghai
```

Figure 57:Sha-1 Hashes 7

Hash Code = SHA1

```
./hashcat-cli32.bin -m 100 --remove hashes-11.raw-sha1.txt
dictionary/passwords.txt/Wikipedia-wordlist.dic -r rules/ T0XlC.rule -o
bestToxlcsha1wikinew.txt
```

```
1.  f1ecaf926c615ba9083fe839b179ca0eedab51c8:Anastassiadis
2.  0820f146e131cd01f9415225f08056b6660749f4:mAndelfingen
3.  bdd8c1113d5b5fa94686bd0011d338f371196815:aksobhyam
4.  03db3886636d89ee920a59f2bd0b731fa22e5e27:Bunapar
5.  2a67a7ecd91017ca3ffb4a3760e5ce37be876637:Yphantes
6.  67823d85d5de91ee649c93384f760d94e59538e2:bolins777
7.  d00a493c26f48bc653d86dc628d902b8d8701adf:excedera.
8.  816d7cb9fabaeb1d041f535b9881f435b2ff9dba:charabaty
9.  5d72158ceb7034211b1b2c71443b29ec580b5cbf:charcutera|
10. 0f8208c33d85c86c19ed2a574e586e4c6e4dcb7e:Oekonomou
11. f123ca567a1b86af20009db6b0dfdd7aae9bc063:Cocufier6
12. 16aad28829e426a49a3ad5fe34a86eed2e458b3b:codame555
13. e6c892a434458be7297880d83ae802b547df3d68:Gorisiez8
14. 058c0437eb04d617f8e39c493bbc6a8946c01915:gouliardi
15. 29a9b7cb6a3c5dd63e40a7cf6728bba9a67f3b44:shassaram
```

Figure 58:Sha-1 Hashes 8

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Hash Code = SHA1

```
./hashcat-cli32.bin -m 100 --remove hashes-11.raw-sha1.txt
dictionary/output.txt -r rules/best64.rule -o best64outputdbsha1.txt
```

```
1.  f835c7819b91737289260db449eaeff698186985:146830822
2.  5b18af45b25033801d58748fa78e108541f90fd8:029748601
3.  467dfb30c1b33b6120adbf8b6b2ddaf4a6735f45:117484222
4.  4606a8a0fd541b7397459572c6626f5992a85d46:044188039
5.  6283f4af6720cfa9a44832f8e0ffbc5047717e57:165906112
6.  224cf26817533a7c7dd90486b2561e9146d73a4a:077866312
7.  d54944bb72528e03d886196fec20f640d4b77f36:234870622
8.  ff5f24855e40288e68b35808b8a5dfa2da9aa020:199uui274
9.  a3bac39f4c0cc2a07c6e41e49e9c34f29ada1773:238783502
10. 426d216bfdfb8ab3293f55f519a83fdf4028daff:32cap2812
11. c0652afcc77478351666eb9ec3dbee66713f28ea:214376913
12. 1c73b6f7418f669af0e3fe7999397e90ef1255fa:283732711
13. b752a2538cf044edb4a14645811d29aa80d0de3c:74243288R
14. e6e49feae8730b47d4c2bd0abccb02107c1a69fa:846600499
15. 1dda3ec23b353e7febed6bccb850fc3fb0d103e4:ANDEREGG3
```

Figure 59:Sha-1 Hashes 10

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Hash Code = SHA1

```
./hashcat-cli32.bin -m 100 --remove hashes-11.raw-sha1.txt
dictionary/naxxatoe-dict-total-new-unsorted.txt -r rules/best64.rule -o
best64naxxatoesha1out.txt
```

```
1.  27594c8a0866b4724f48dd0afbdc8fdbdc6177bc:I don’t see why
2.  4bd13a307d126d43946c344dac39cc090328e066:for a short time
3.  684bd240578f87ef08298d2a248502ead1f8d08d:in your power
4.  67f8f39564802777c833c037107926e577e76576:What do you know
5.  903044dc848f5f0128ae15ecad703dd14510e85f:yvridikoi
6.  01fb5503b2ec1048fab936f5d1fe51837f15571d:S’il vous plait
7.  1a20a1c38da396d0d5aaae3ce4efe5f2ae27e721:b!rrett45
8.  30fd234c4426dda163b8db9055c554565b22450e:go1dene57
9.  efcfd8a8016176a11dc2ec1c727fb0b557ab2655:Fr3$c01ng
10. 9e54d6d5055b14101ef67d169d20a057c279047f:M4rr14g35
11. 7f6f403875cc0dadb5a3c6da00cfaaeb8de3b03b:pos!7!ons
12. c94eded20c4df2bbc7408b200412fb5c119fb996:REV0LUT10N$
13. 724dfcceb4fdb01d1aa9a280b4df8b8796a1fab4:Murali5+5
14. 208637dfa9142f93e6e67aabcd6f11efa560b79f:2 YOUSSEF
15. 5d6ddee5b2ae32706aea9c45cef059de35b2e6ae:rashivers
16. 6efacad86aadb46d74217ab8cbcd500bd467fada:plutolock
```

Figure 60:Sha-1 Hashes 11

Summary of the attack on SHA1 Hashes

Wordlist name	Rule	Result
Passwords.txt	Best64	426
Output.txt	Best64	81
wikipedia-wordlist-sraveau-20090325	Best64	52
naxxatoe-dict-total-new-unsorted.txt	Best64	16
Passwords.txt	Specific	73
-do-	Toxlc	216
-do-	Generated	321
-do-	Combinator	11
-do-	Leetspeak	1
wikipedia-wordlist-sraveau-20090325	Specific	11
-do-	Toxlc	38
-do-	Generated	1
Grand Total	**--**	**1247**

Figure 61:Summary on Sha-1 Hashes

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

4.2 JTR

Hash Code = DES
./john hashes-3.des.txt
1. zzgDEnyTADkggw:RonThomp
2. IMyXn8qsuEBEo:password
3. s7ZGhJAo95euw:mountain
4. /sEpoQ4fV.xE2:superman
5. 5CEMOQdm503Yg:chocolat
6. cSpCacFaJ70Cg:cocacola
7. jYKki.hzO1i4I:nicholas
8. /3Oeeb.AxlWOg:snoopdog
9. xUdPseSg/OgpU:remember
10. j9FlR1tGMPprg:starwars
11. QETeGia5g1ZNE:beatrice
12. PMPlzZ8xf9pJA:harrison
13. 1/.yeOT4sp.i2:Elizabet
14. aoB5Ek0H91GXA:anything
15. D9n8wrD/umZXc:future

Figure 62:Jtr

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

4.3 Passwords Pro

Hash Code = NTLM
Passwordspro.exe
1. c0e2b0ee40bdaeacdf6982728ca8244b:!Portland
2. 633bc80a9e8ad463ed6ccdfd500d4484:$pears.o6
3. 07c9dc1fbadd32693deee19767afcdda:024546189
4. 9659504e0d44c42daa595673cc7ccc6e:050670537
5. aa2860ae3079abb6ac79a2ea645c33cb:06012118t
6. e2121e28ac7e2642300a3839d80827de:065470255
7. b713558e994ad30ee1e37a37467effeb:07-06-1683
8. 752a73192d06093ffadc09e2dc6c24ad:07.09.1798
9. f87c9d38794ce27e52550f41353c2028:070750672
10. 615eee058c33773830a7c7e9f5d42a04:10AUG1769
11. 32fc097a03be387f366909fb08e796be:10Feb1111
12. 2f493f5e5f6f305402a682b7f02e9d31:11-02-1319
13. ac62fe2a3c729bc43e78a95cc97bdb1b:11apr1315
14. 077251a1ab2f184602e33aec85848a0c:130617___
15. 43f1f62909b41fd5fd60f9e4ef830c3e:13aug1457

Figure 63:PasswordsPro

4.4 Ophcrack

Hash Code = NTLM Crack using Rainbow Tables

Ophcrack

```
1.  reyesco:51::c038be0198418ab29dc9c3463d9e8151:::Kosti
2.  simpsonk:61::f6f7f9425701e9015bce574873f042ec:::mission43
3.  dianar:182::5cdd56caaca56cd0876fa8118ffc615e:::copper101
4.  alyss.watson:247::698b75626b04c9f82cb1af59aa67a9c5:::xmichal11
5.  melis.perez:310::e5a7c364e86bb94655aab819eca4343e:::belize321
6.  darichards:325::24482b9b25c7443772704136e4ee4521:::paulina891
7.  johnsogi:376::b1e1f54e911e8975afa6c2b42f495413:::interview
8.  gonzalma:437::352cca4675e3d2cdb183067582f2e840:::clarice95
9.  alexi.gonzalez:454::30f58859559639b12613fe5c9992a6c4:::Routso
10. dagarcia:529::8a6995d6f1f32eecaf2312f7f310832d:::Julas
11. amanda:549::0407765c188bf6e87ee8450ffd2baabc:::know
12. phills:564::1b613a1bfa938200cda55392199261af:::orbitalco
13. bill.kumar:728::b1b6b3a527d485c05de708e208ece413:::!dillinger16
14. sawilliams:758::d25de3cb8a21da50f2e3ab338180ed3e:::Zoes
15. sharmaad:818::06c81f4f787d49609371497d746f2bf9:::leonard88
```

Figure 64:Ophcrack

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

5. RESULTS AND RESEARCH THROUGH COMPARISONS

Although there are countless tools available in the market for cracking almost all kind of Hashes, four tools there was chosen that may be called as the market leaders in this regard. These tools are hashcat, John the Ripper, Passwordspro and the Ophcrack. All these tools were used on a 32 bit Linux Interface. The hashcat and John the Ripper were used in a command line interface. The Passwordspro and the Ophcrack's Graphical Front End were utilized. All the tests were conducted using NT hashes supplied during the DEFCON 2012 Competition. The file hashes-7.nt contains almost 15317 hashes in it. The Hashcat, John the Ripper and the Passwordspro were used to initiate a simple dictionary based attacks, while Ophcrack supports RainBow Table Attack methodology only. So ophcrack was used only to crack NT hashes using tables. All the results are fully dependent on the type of hashes provided for this comparison.

The following Graphs provide a total overview on what each tool has managed to do.

5.1 HASHCAT

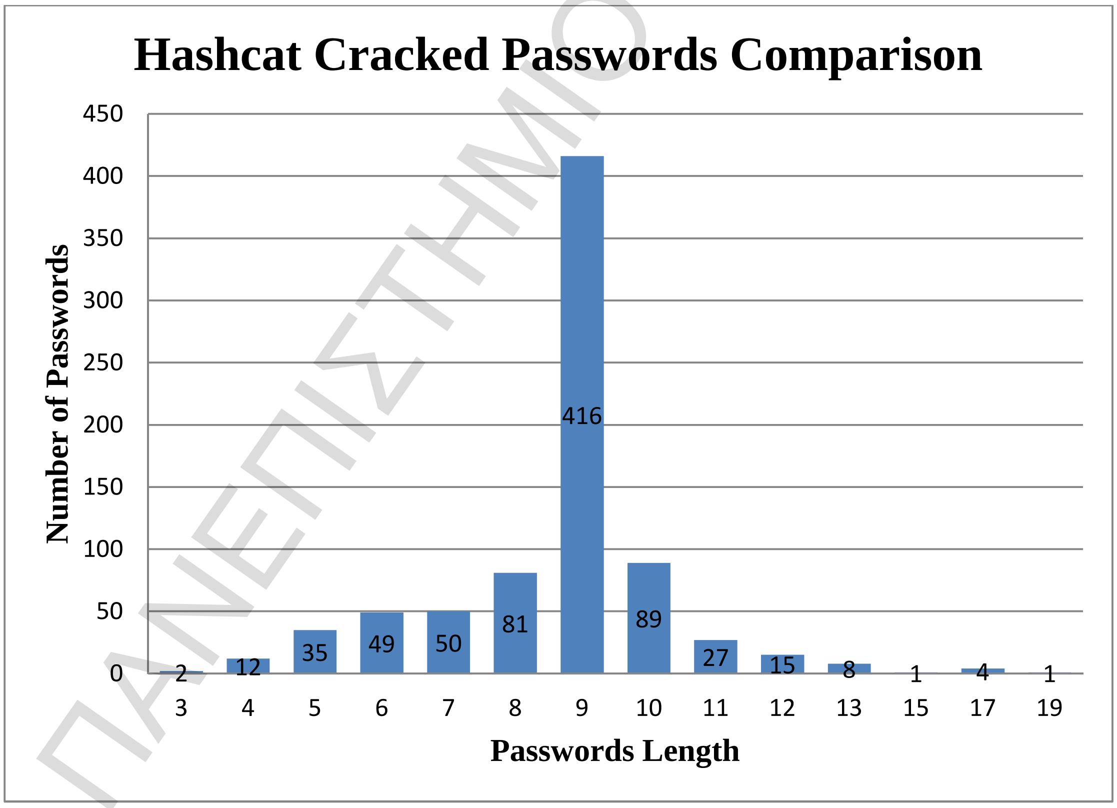


Figure 65:- NT hashes Cracked Passwords Length Coverage.

This graph clearly shows that the passwords having length of nine characters were cracked mostly. Another fact is that hashcat can crack lengthy passwords. It cracked a password that was nineteen characters long. The upcoming graphs of other tools clearly shown that no other tool can surpass this level of performance.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

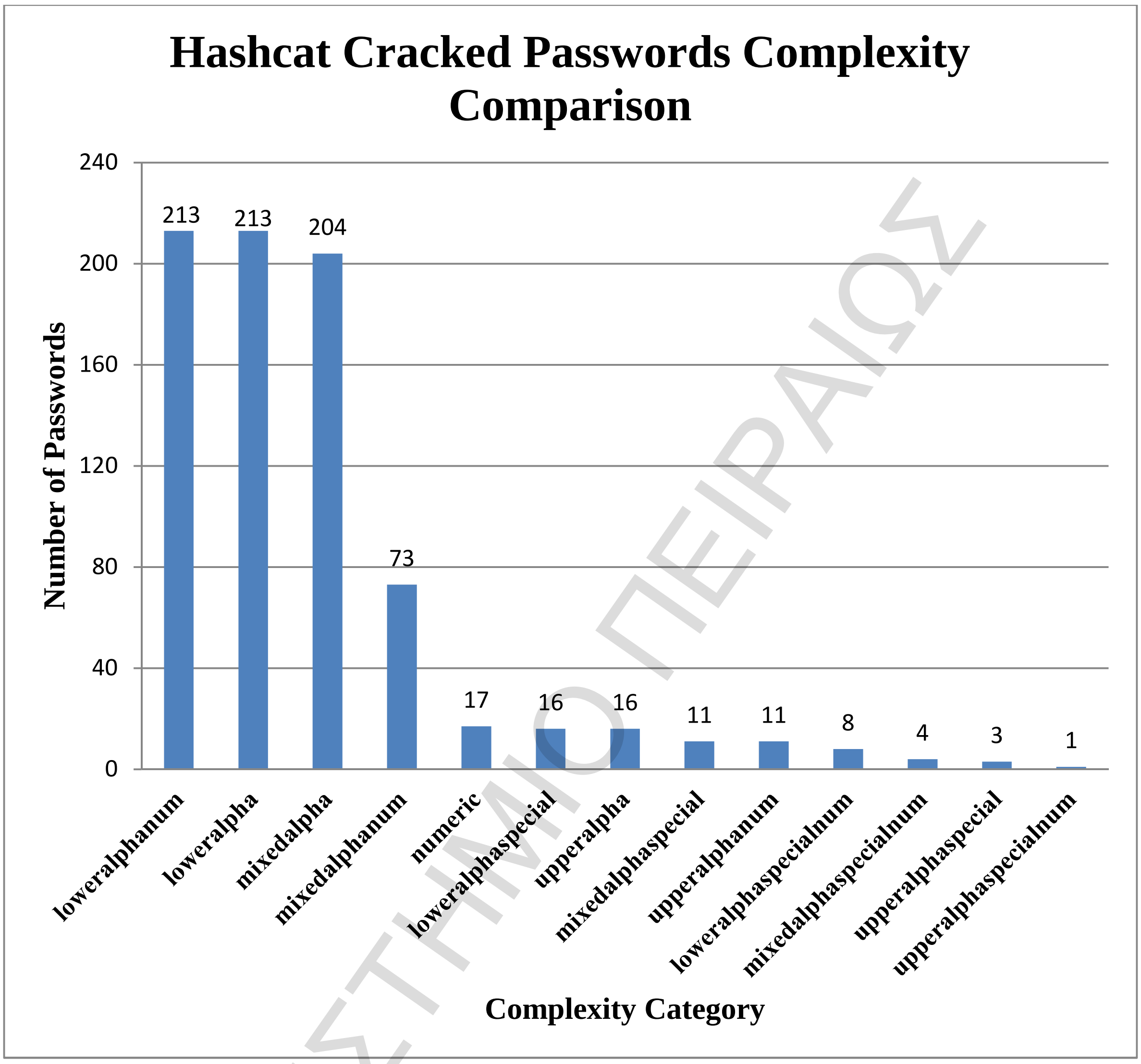


Figure 66:- Hashcat Cracked Passwords Complexity Paradigm

This graph shows that hashcat mostly cracked passwords of lower alpha number, lower alpha, mixed alpha and mixed alpha number types. There is another important characteristic of hashcat revealed and that is the fact that it cracked passwords of thirteen different types.

Now the Following Graphs Shows the Performance matrix of the famous John the Ripper.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

5.2 JTR

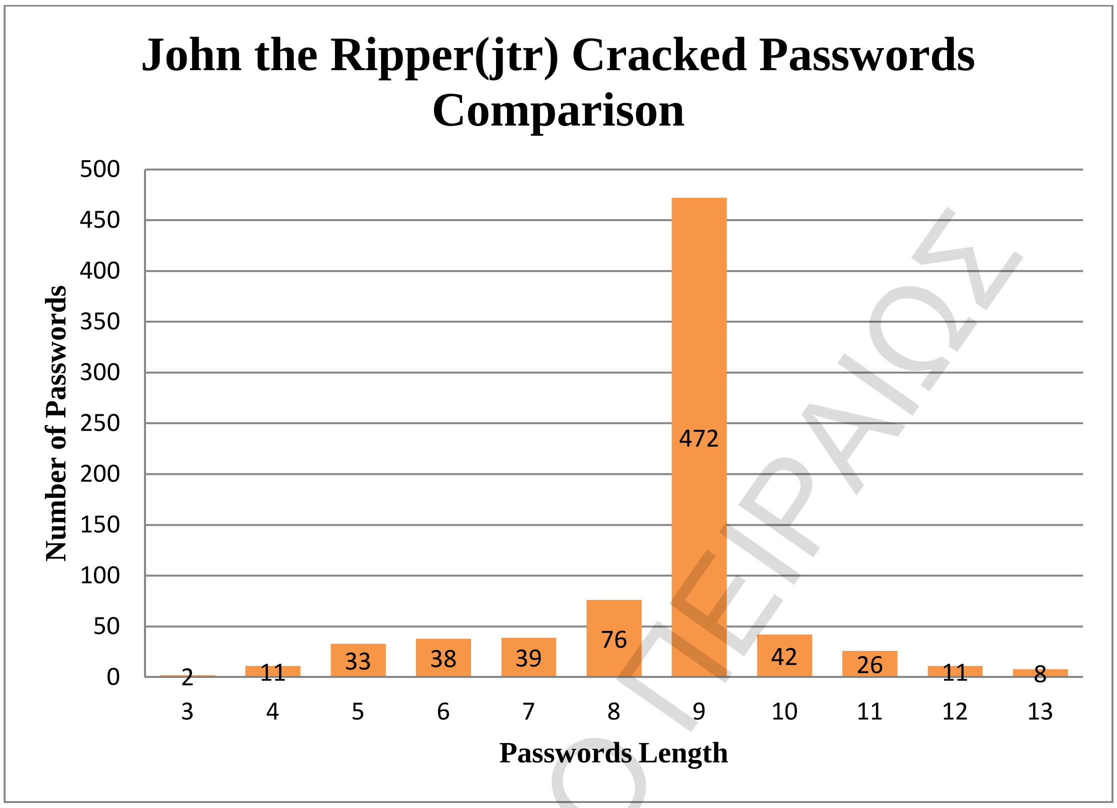


Figure 67:- John the Ripper Cracked Passwords Length Coverage.

In the above comparison John the Ripper shows its performance peak in breaking nine characters long passwords more than any other length of passwords.

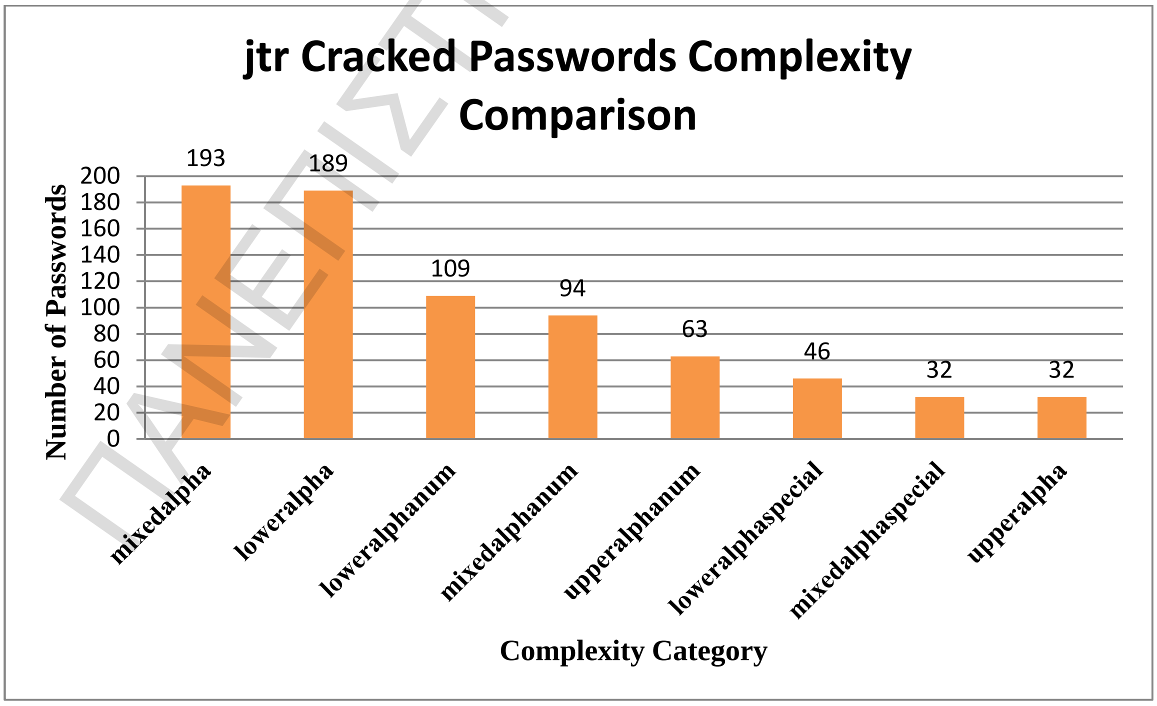


Figure 68:- John the Ripper Cracked Passwords Complexity Spectrum.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

John the Ripper cracked passwords analysis shows that it can crack passwords having complexity of eight different characteristics.

The following Graphs shows the performance of the GUI based Tools.

5.3 PASSWORDS PRO

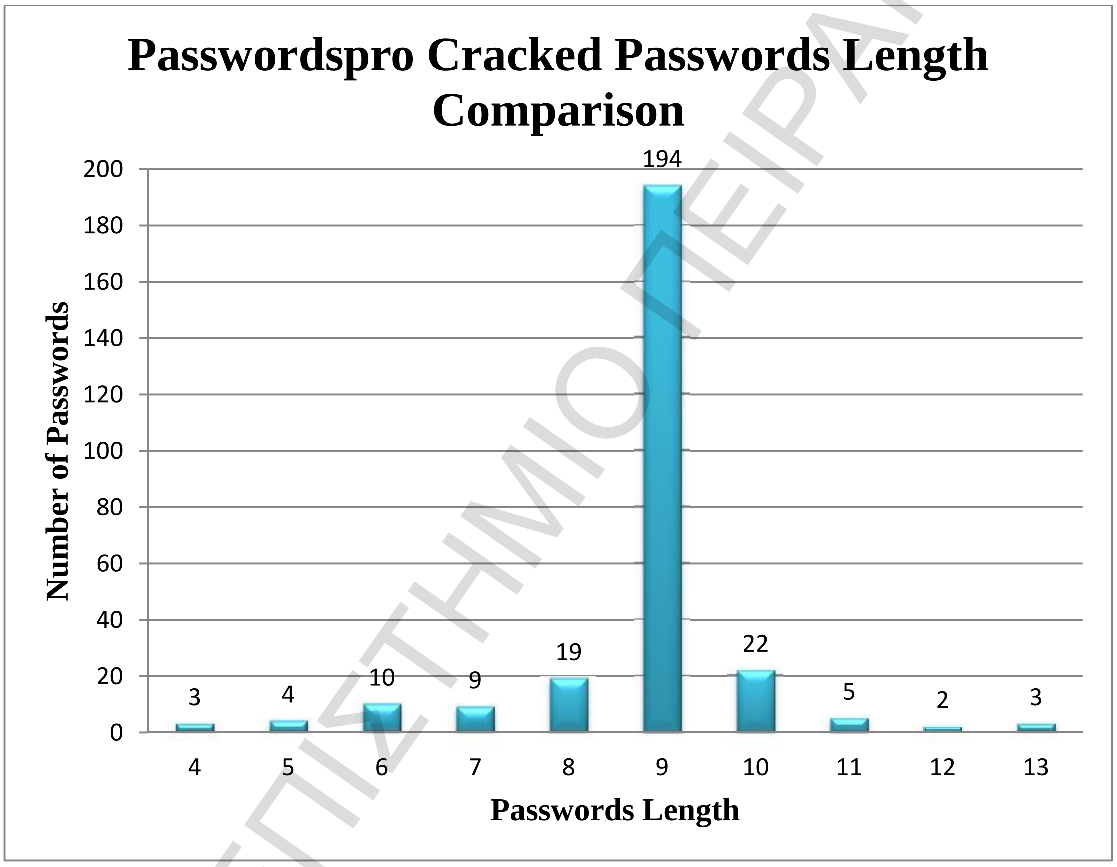


Figure 69:- Passwordspro gains the Passwords length score

In this comparison, again it is clear that InsidePro's PasswordsPro also cracked mostly nine length passwords. It also cracked thirteen characters long passwords at most when cracking NT hashes.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

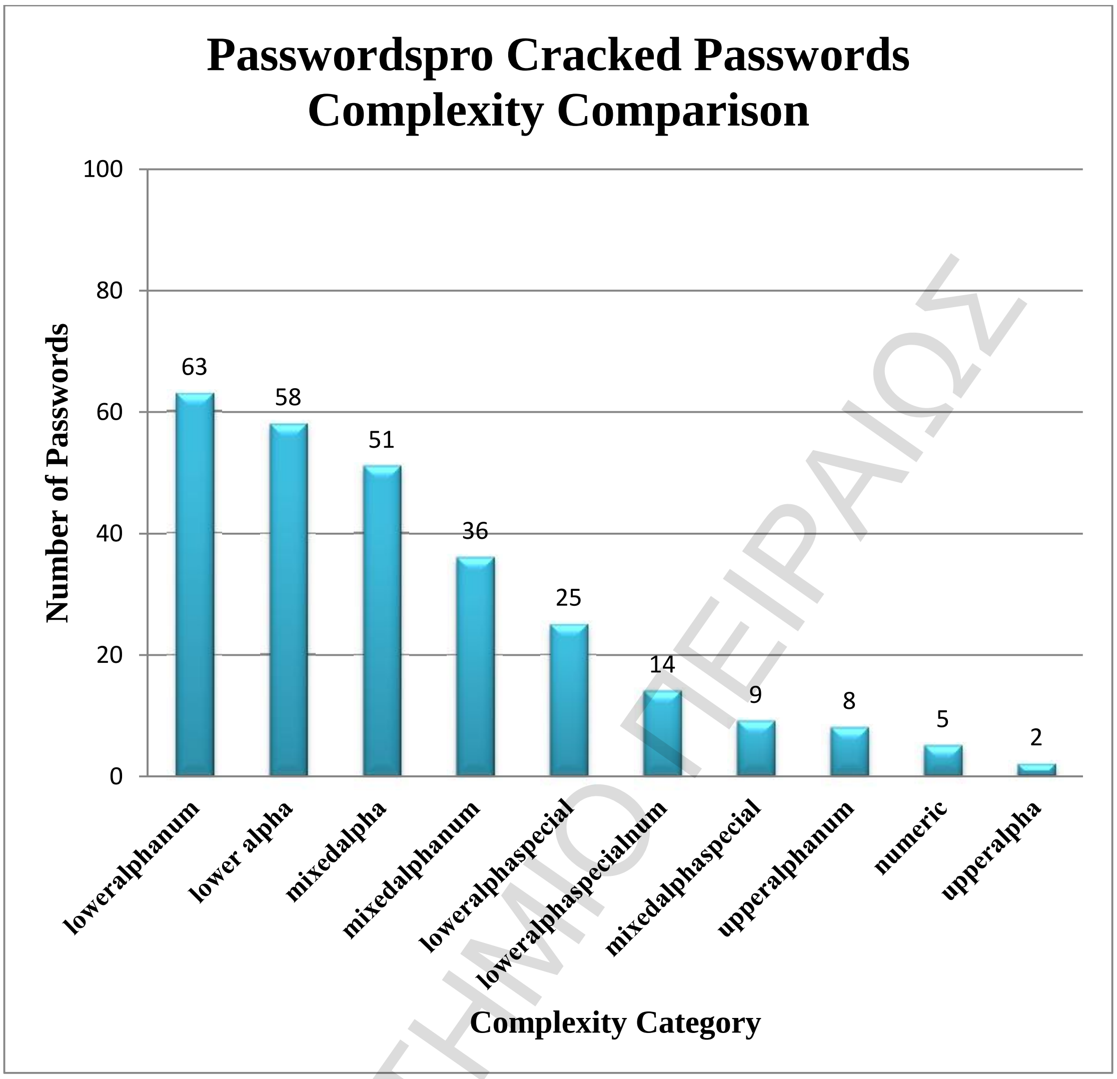


Figure 70:- PasswordsPro cracked passwords Complexity

PasswordsPro cracked passwords having complexity score of ten types. It cracked lower alpha number; lower alpha mixed alpha num types and the rest of in order.

The upcoming graphs show the results that came by ophcrack using rainbow tables.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

5.4 OPHCRACK

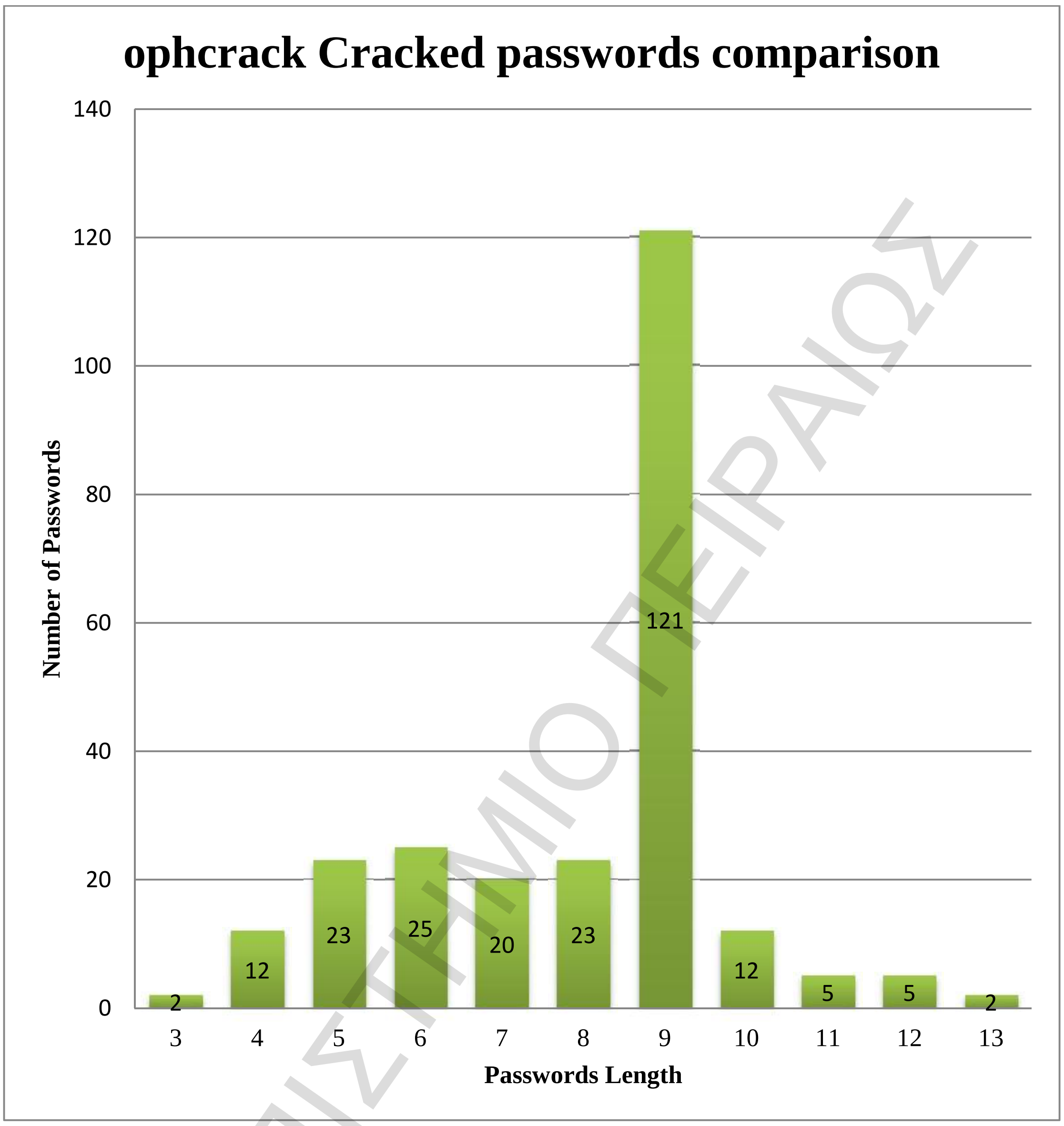


Figure 71:- ophcrack Rainbow Tables Cracked Passwords Length.

Ophcrack broke nine characters long passwords more than often. It cracked no bigger than thirteen characters long passwords.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

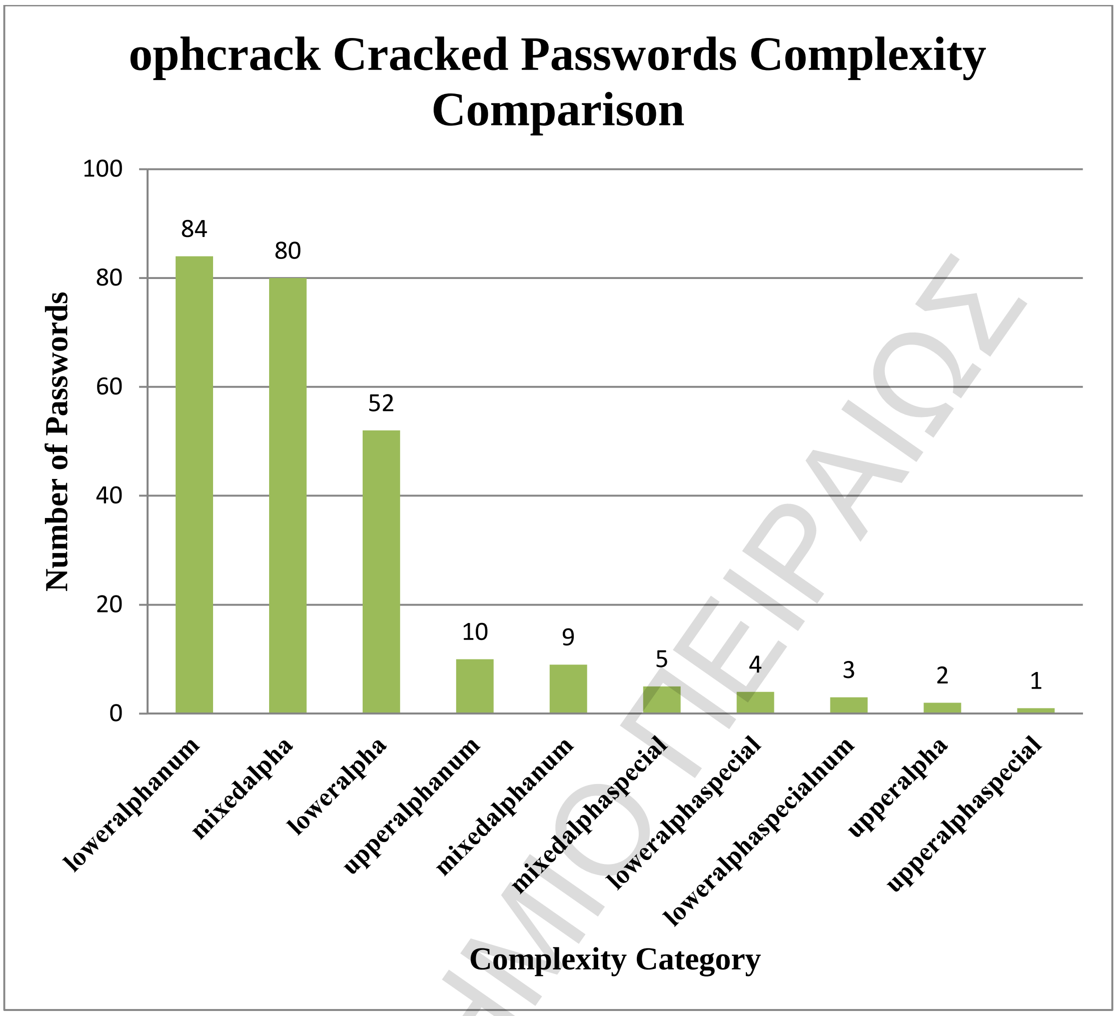


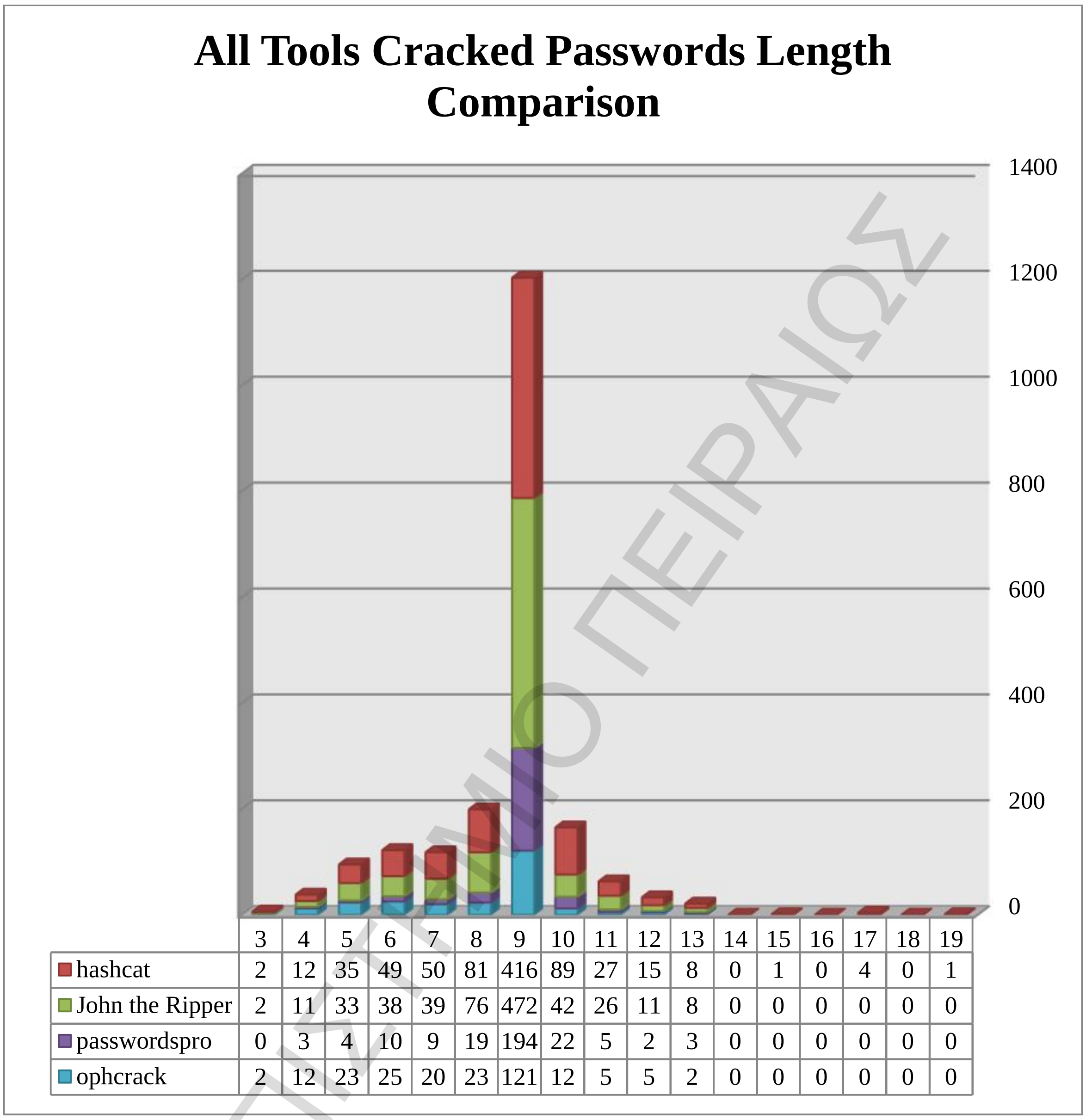
Figure 72:- ophcrack cracked passwords Complexity score.

Ophcrack cracked passwords complexity category score was ten. Mostly, it cracked lower alpha num, mixed alpha, lower alpha and upper alpha num types passwords.

The upcoming Graphs show an effort of making a combined analysis of all these four tools , Hashcat, John the Ripper ,PasswordsPro and ophcrack.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

5.5 All Tools Comparison

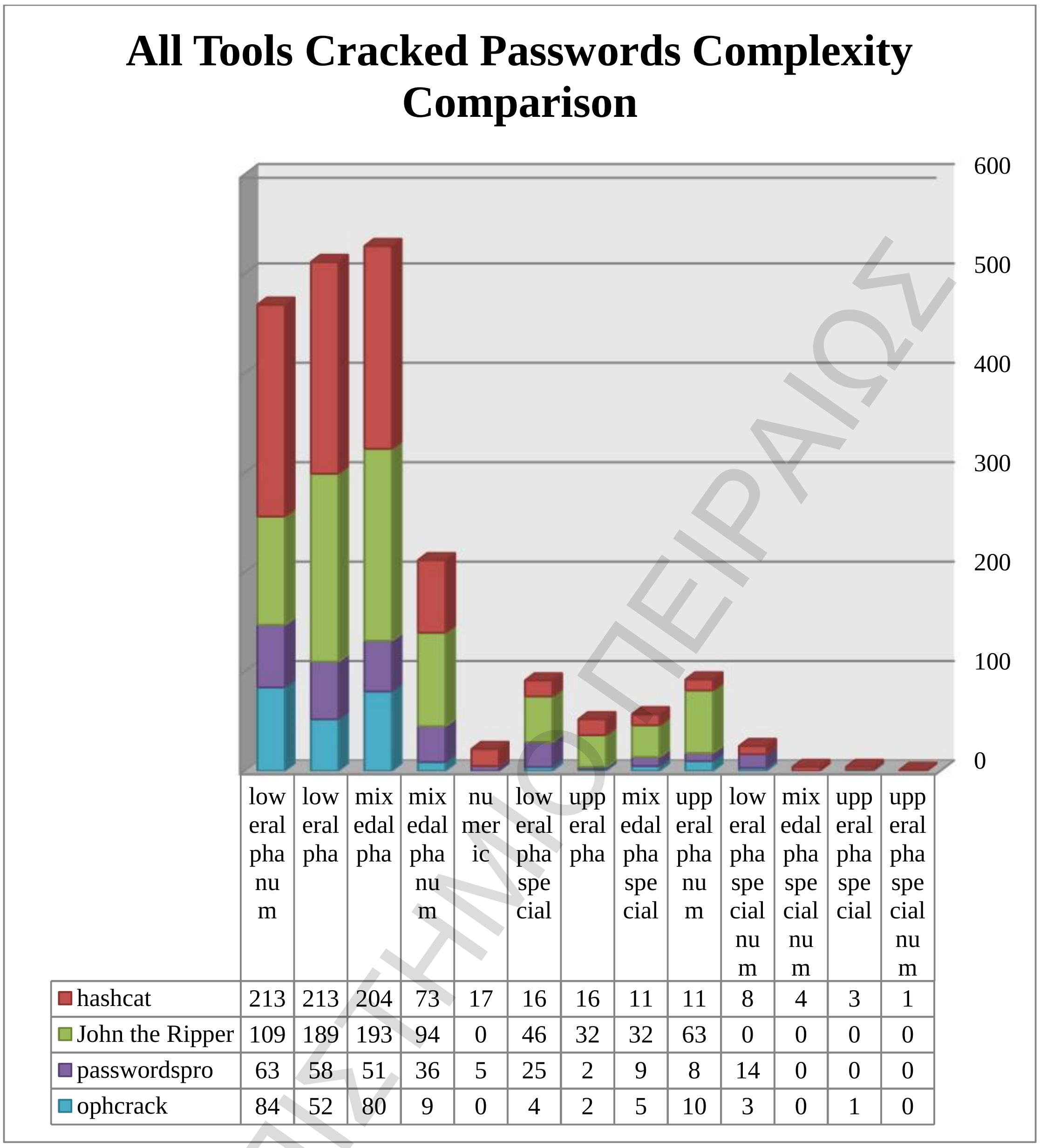


	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
🟥 hashcat	2	12	35	49	50	81	416	89	27	15	8	0	1	0	4	0	1
🟩 John the Ripper	2	11	33	38	39	76	472	42	26	11	8	0	0	0	0	0	0
🟪 passwordspro	0	3	4	10	9	19	194	22	5	2	3	0	0	0	0	0	0
🟦 ophcrack	2	12	23	25	20	23	121	12	5	5	2	0	0	0	0	0	0

Figure 73:- All Tools Cracked Password's Length Spectrum.

In the above graph, all tools mostly crack nine characters long passwords. John the Ripper tool is ranked on top according to this analysis. But when talking about the coverage of the password's length hashcat win the race by cracking a nineteen characters long password.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ



	loweralphanum	loweralpha	mixedalpha	mixedalphanum	numeric	loweralphaspecial	upperalpha	mixedalphaspecial	upperalphanum	loweralphaspecialnum	mixedalphaspecialnum	upperalphaspecial	upperalphaspecialnum
🟥 hashcat	213	213	204	73	17	16	16	11	11	8	4	3	1
🟩 John the Ripper	109	189	193	94	0	46	32	32	63	0	0	0	0
🟪 passwordspro	63	58	51	36	5	25	2	9	8	14	0	0	0
🟦 ophcrack	84	52	80	9	0	4	2	5	10	3	0	1	0

Figure 74:- All Tools Cracked Passwords Complexity Comparison.

In the above graph, it is more than obvious that hashcat cover a lot of complexity categories as compared to other tools. Surprisingly, john the ripper covered only eight types of complexity such a low score in this survey.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

5.6 Summary of Comparison

Summarizing what the graphs unveiled it is important to point out that all the comparisons were made only on NT hashes, because all the four tools chosen support this hash type. So if the input of hash type did change, the result may vary substantially. It is obvious from these graphs that no one's password can be called as 'un-crackable'. When looking at Figure 73, surprisingly, all the four tools "like" to crack passwords having a length of nine characters. Another important aspect of this analysis is that when Figure 74 is closely examined, it becomes straightforward the hashcat cracked password complexity spectrum. It checked the hashes and compares them against thirteen different complexity categories. It is simply enough and it seems that nothing is left behind.

A strong result is made after the password cracking exhausting procedure, that hashcat is a superb tool for cracking. It can crack many types of Hashes with a given high level of complexity at a fairly rapid rate. But John the Ripper surpasses when comparing the DES hashes cracking performance of both tools. Hashcat consumes all the processing power of the CPU currently in hand, while the John the Ripper does never hold too much pressure on the CPU cores.

This situation leads toward the conclusion that hashcat is faster than John the Ripper. There are lots of professionals that like to use both tools. Passwordspro takes the third position because it is not as much successful as the hashcat and John the Ripper. It is less fast as compared to both Command Line tools. It is also closed source software. Ophcrack takes the last position because it's rainbow tables was least effective. The given hashes sample set was quite large enough for the table based processing. Ophcrack takes almost ten hours of intensive CPU utilization to crack only two hundred and fifty NT hashes. And it should not be forgotten that Ophcrack only supports NT hashes and rainbow table's methodology in order to crack hashes.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

5.7 Conclusion and Suggestions

From a hackers perspective, (13) a users password can be made secure by educating the user itself. They must learn that a hacker or cracker can break his password when they are online using a website or offline by cracking their root password. To avoid offline attack they need a secure hash type to store passwords (2). A hash type that is totally unfriendly for the CPU or GPU hardware or software, very unfriendly for brute force or dictionary attacks. Unfriendly hash type means a hash that takes days or weeks to break. The user need strong and strict password policy implemented at their end like avoiding the use of same password across multiple sites.

When talking about the online attack threat model they must learn that there should be a strong policy that checks the users selection of very common passwords spread over the internet. There are lot of threats that can be counted here like server vulnerability exploits ,network based attacks and attacks targeted towards users by malware, virus and spam softwares. A user must know how to block these efforts made by hackers by using anti viruses, firewalls etc. There will be situations needed to avoid possible loss of user's personal information or data stored in the computer hard drive.

It is the job of the senior management, enterprise level architects and the Senior IT professionals to design Passwords Selection Strategy. Four basic techniques can be used (15). Educate the user about the importance of the passwords selection that is not easy to guess while memorable. Secondly, computer generated passwords can be used but the underlying problem is that they are not easy to remember. Thirdly, reactive passwords checking should be made by the system by cracking the guessable passwords. The fourth approach is then to implement a proactive password checker software. It facilitates the user to select a strong password that is acceptable and memorable for him but hard to guess by cracker softwares.

A rule enforcement system is suggested and famous examples are passwords enforcement policies while a dictionary of banned passwords (16) (17) (18) (19) should also be implemented. There is another important aspect of systems security, the one that says that systems who don't need internet access should not be granted access. It effectively reduces the threat for the user and the organization.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

5.8 Future Work

In this paper there have been an effort to make clear what passwords are, what are the main methods of attacking them, how people can produce strong passwords and that implementation of password policies is very crucial.

Moreover computer enormous progress have been presented and ways of cracking passwords faster have been displayed. Several basic tools has been put in front and thoroughly tested in order to gather some very useful results from the cracking process.

There is left space though to continue this work already done and try even more hashes with the tools described earlier. That way understanding in the field of cracking passwords will get even better and brighter. The ultimate goal though through this paper and the tryouts that will may be carried out in the future is to have greater security not only as far as passwords are concerned but also in the information systems field in general. This will give people that are not security oriented the awareness needed to get through this challenging world and feel a bit safer.

In the end everyone knows that no matter what organizations or people in general do to secure information there is not and will never be 100% security. If people understand that and make it a way of life things will get a bit quitter and their life's a bit easier.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

6. References

(1) http://en.wikipedia.org/wiki/Password_cracking

(2) http://en.wikipedia.org/wiki/Password_strength

(3) http://en.wikipedia.org/wiki/Password_policy

(4) http://en.wikipedia.org/wiki/Salt_(cryptography)

(5) http://en.wikipedia.org/wiki/Key_stretching

(6) http://arstechnica.com/security/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours/

(7) http://www.insidepro.com/eng/passwordspro.shtml

(8) http://www.openwall.com/john/

(9) http://ophcrack.sourceforge.net/

(10) http://hashcat.net/oclhashcat-plus/

(11) http://project-rainbowcrack.com/

(12) http://www.oxid.it/cain.html

(13) John R. Vacca. Computer and Information Security Handbook (Morgan Kaufmann Series in Computer Security) June 2009.

(14) Simon Marechal, Solar Designer, Password security: past, present, future presentation.

http://www.openwall.com/presentations/Passwords12-The-Future-Of-Hashing/

(15) William Stallings cryptography and network security principles and practices, Third edition Aug 2002.

(16) Architecting Security, Password Patterns

http://www.architectingsecurity.com/2010/09/11/password-patterns/

(17) Most common passwords list from 3 databases

http://blog.jimmyr.com/Password_analysis_of_databases_that_were_hacked_28_2009.php

(18) A Survey of Password Attacks and Comparative Analysis on Methods for secure Authentication, Mudassar Raza, Muhammad Iqbal, Muhammad Sharif and Waqas Haider. World Applied Sciences Journal 19 (4): 439-444, 2012 ISSN 1818-4952; © IDOSI Publications, 2012 DOI: 10.5829/idosi.wasj.2012.19.04.1837

(19) Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords Matt Weir, Sudhir Aggarwal, Michael Collins, Henry Stern Florida State University, Redjack LLC, and Cisco IronPort Systems.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

7.Appendix

Wordlist collections
http://home.btconnect.com/md5decrypter/passwords.zip
http://home.btconnect.com/md5decrypter/output.rar
http://contest-2010.korelogic.com/wordlists.html
http://www.skullsecurity.org/wiki/index.php/Passwords
http://www.openwall.com/wordlists/
http://www.insidepro.com/dictionaries.php
http://ophcrack.sourceforge.net/tables.php

Figure 75:Wordlists Collection