

AN INVESTIGATION OF THE SECURITY OF
PASSWORDS DERIVED FROM AFRICAN
LANGUAGES

Submitted in partial fulfilment
for the requirements of the degree of

MASTER OF SCIENCE

of Rhodes University

Sibusiso Sishi

Grahamstown, South Africa

March 2019

Abstract

Password authentication has become ubiquitous in the cyber age. To-date, there have been several studies on country based passwords by authors who studied, amongst others, English, Finnish, Italian and Chinese based passwords. However, there has been a lack of focused study on the type of passwords that are being created in Africa and whether there are benefits in creating passwords in an African language.

For this research, password databases containing LAN Manager (LM) and NT LAN Manager (NTLM) hashes extracted from South African organisations in a variety of sectors in the economy, were obtained to gain an understanding of user behaviour in creating passwords. Analysis of the passwords obtained from these hashes (using several cracking methods) showed that many organisational passwords are based on the English language. This is understandable considering that the business language in South Africa is English even though South Africa has 11 official languages.

African language based passwords were derived from known English weak passwords and some of the passwords were appended with numbers and special characters. The African based passwords created using eight Southern African languages were then uploaded to the Internet to test the security around using passwords based on African languages. Since most of the passwords were able to be cracked by third party researchers, we conclude that any password that is derived from known weak English words marked no improvement in the security of a password written in an African language, especially the more widely spoken languages, namely, isiZulu, isiXhosa and Setswana.

Acknowledgement

I would like to thank my supervisor Professor K. Bradshaw for all the help and guidance with this research paper and to my family for being the foundation.

ACM Computing Classification System Classification

- **Security and privacy ~ Authentication**
- Security and privacy ~ Multi-factor authentication
- Security and privacy ~ Usability in security and privacy

Glossary

AD Active Directory. 3, 6, 8, 21, 22, 23, 24, 25, 30, 39, 43, 44, 58, 74

API Application Programming Interface. 23

ASCII American Standard Code for Information Interchange. 9, 10

CIS Center for Internet Security. 8, 9, 10, 13, 29, 58

CPU Central Processing Unit. 35

CSV Comma-Separated Values. 42, 49

DES Data Encryption Standard. 24, 25, 51

DSA Directory System Agent. 23

ESE Extensible Storage Engine. 21, 23

GPU Graphics Processing Unit. 3, 35, 36

IT information technology. 1, 6, 11, 13, 15, 16

LDAP Lightweight Directory Access Protocol. 22, 23, 24

LM LAN Manager. viii, x, xii, 4, 24, 25, 30, 31, 32, 35, 36, 37, 38, 39, 40, 42, 43, 44, 45, 44, 45, 46, 47, 48, 49, 50, 51, 74

MD4 Message Digest 4. 25

NIST National Institute of Standards and Technology. 3, 9, 10, 13, 14, 29, 58, 74

NTLM NT LAN Manager. viii, x, xii, 3, 4, 8, 24, 25, 30, 31, 32, 35, 36, 37, 38, 39, 40, 41, 42, 45, 47, 48, 49, 50, 51, 58, 68, 70

SASL Simple Authentication and Security Layer. 24

SHA1 Secure Hash Algorithm 1. 17

SID Security ID. 38, 42

SMS Short Message Service. 10

SSL Secure Sockets Layer. 24

TLS Transport Layer Security. 24

Contents

Glossary	iv
List of Figures	xi
List of Tables	xiii
1 Introduction	1
1.1 Context of the research	1
1.2 Research statement	2
1.3 Research objectives	3
1.4 Approach	3
1.5 Limitations	4
1.6 Document structure	4
2 Literature review	6
2.1 Overview of passwords	6
2.2 Organisational password policy	7
2.2.1 Typical South African password policies	10
2.2.2 Password policy concerns	14

2.2.3	Selection of passwords	15
2.3	Password attack methods	16
2.3.1	Attacks using dictionaries	17
2.3.2	Brute-force attacks	17
2.3.3	Attacks using rainbow tables	18
2.3.4	Attacks using part-of-speech tagging	18
2.3.5	Keyboard walking	19
2.4	Microsoft active directory database	21
2.4.1	Password storage	23
2.4.2	Windows password hashing algorithms	24
2.5	Analysis of Chinese regional breached passwords	26
2.6	Summary	29
3	Research design	31
3.1	High-level overview of research	31
3.2	Password databases	32
3.3	Breached password lists	33
3.3.1	HaveIbeenPwned	33
3.3.2	Breachcompilation	33
3.3.3	Weakpass_2a	33
3.4	Rules used	34
3.5	Creating African language derived passwords	34
3.6	Hardware/software configuration	36
3.7	Hashcat attack modes	36
3.8	Summary	37

4	Analysis of LM and NTLM hashes	38
4.1	Extraction software	38
4.2	Extraction of hashes	41
4.3	LAN Manager (LM) hashes	43
4.4	NT LAN Manager (NTLM) hashes	46
4.5	LM hash attack	47
4.6	NTLM hash attack	48
4.7	Characters for LM and NTLM hashes	49
4.8	Summary	50
5	Analysis and discussion of organisational passwords	51
5.1	Cracked passwords	51
5.2	Top ten passwords	52
5.3	Top ten base words	52
5.4	Keyboard walking patterns	55
5.5	Password length analysis	58
5.6	Analysis of digits used	60
5.7	Use of characters in passwords	61
5.8	Letter frequency analysis	62
5.9	Character set usage	64
5.10	Summary	65

6	Generating and analysing passwords using African languages	66
6.1	Creation of African language passwords	66
6.2	Further processing of passwords generated	69
6.3	Analysis of African derived passwords	70
6.3.1	African language base words found	71
6.3.2	Passwords found	72
6.4	Summary	73
7	Conclusion and future work	74
7.1	Summary	74
7.2	Future work	75

List of Figures

2.1	Password examples taken from CapeNature (2007)	13
2.2	Example of keyboard walking	20
2.3	Second example of keyboard walking	20
2.4	Third example of keyboard walking	20
2.5	Square keyboard walking	20
2.6	Directional keyboard walking	21
2.7	Rectanglular east and west pattern	21
2.8	Physical components of the Data store taken from Microsoft (2014)	22
2.9	Data store architecture taken from Microsoft (2014)	23
2.10	LM hash creation from Johansson (n.d.)	25
2.11	NTLM hash creation from Johansson (n.d.)	26
2.12	Top weak Chinese and English characters obtained from Li <i>et al.</i> (2014)	27
4.1	Extract of Impacket’s secretsdump	38
4.2	NTDS.dit, SYSTEM and SECURITY extraction using <i>NTDSUtil</i>	40
4.3	NTDS.dit and SYSTEM extraction using <i>Vssadmin</i>	41
4.4	LM hash treemap top 100	44

4.5	LM hash top ten	45
4.6	NTLM hash top 20	46
5.1	Cracking passwords using keyboard walking	56
5.2	Keypad walking patterns used	57
5.3	Top 20 character counts in cracked password list	61
5.4	Top 20 character heatmap where darker colours show higher frequency of use	62
5.5	English letter frequency	63
5.6	Cracked password letter frequency	63
6.1	Venda special characters	66

List of Tables

2.1	Li <i>et al.</i> (2014) leaked password datasets	26
2.2	Top weak Chinese and English passwords obtained from Li <i>et al.</i> (2014) . .	27
2.3	Top five Chinese Pinyins and English words obtained from Li <i>et al.</i> (2014)	28
2.4	Eight character date formats used obtained from Li <i>et al.</i> (2014)	29
2.5	Six character dates used obtained from Li <i>et al.</i> (2014)	29
3.1	Hardware used to extract and analyse the NTLM and LM hashes	36
3.2	Software used to obtain clear text passwords	36
4.1	Hashes collected	42
4.2	Count of least frequently appearing LM hashes	45
4.3	Count of least frequently appearing NTLM hashes	47
4.4	NTLM passwords found using wordlists and rules	48
5.1	Top ten passwords found	52
5.2	Top ten base words	53
5.3	Top ten password lengths	59
5.4	Sequences of digits appended to passwords	60
5.5	Character sets for cracked password list	65

6.1	Top weak passwords per language	67
6.2	Months represented in the various languages	68
6.3	Generated password lengths	70
6.4	Cracked words found per language	71
6.5	Analysis of words depicting months found	72
6.6	Cracked password lengths	72

Chapter 1

Introduction

1.1 Context of the research

While every network penetration tester has his/her own method of testing the internal security of an information technology (IT) environment, the methods used are never the same. "*The things that remain constant however, are that users are always the weakest link*"(Adams & Sasse, 1999; Sasse *et al.*, 2001) and the passwords they create within the organisation are normally poorly designed. Organisations spend millions on their security, from patching management solutions, installing next generation firewalls and having intrusion detect on systems. However, the one weakest link in any cyber security system is the end user with the default *Abc1234567* password.

When IT administrators, users and executives of organisations create passwords, there is a trade-off between usability and security (Dell'Amico *et al.*, 2010). While strong passwords are harder to guess and attack they can also be difficult to remember for the end user. In light of the usability versus security tradeoff, users often knowingly choose to use weak passwords or circumvent security best practices in order to use the password that they have in mind.

As passwords are used on a daily basis by everyone to gain access to systems and applications, understanding which passwords are being used within the organisation is important as a weak password can be abused to bypass security controls (Whittaker, 2015). Organisations are being breached on a daily basis (Gemalto, 2017) and the Internet contains millions of passwords that are being used by users around the world. A quick assessment

of some of these breached passwords, reveals that many of these are based on the English language.

There are various ways that a password hash can be obtained from an organisation. For example if the organisation's Active Directory is compromised the hashes stored in the database can be extracted and attacked offline. Using password lists from various websites such as *Have I been pwned*, *Insiderpro*, *Hashkiller*, *Crackstation*, Daniel Miessler's *Seclist and publicdbhost* to name but a few, allows the general public to download lists that contain weak passwords and previously breached passwords and to attack offline password hashes.

In an ideal world users would create passwords that are long, complex and have enough entropy that an attacker would find it difficult to obtain the clear text password. This may increase the organisation's and the user's cyber security posture.

In South Africa, the business language that people use for normal communication is English and therefore it is assumed that the majority of employees will create English passwords. However, most password lists that are publicly available on the Internet contain many variations of English words, which an attacker can use to try and guess user passwords.

Given that South Africa has eleven official languages (Republic of South Africa, 2011, p.4), it is likely that those whose mother tongue is not English will create passwords that are not based on the English language. This, then, leads to many questions related to the ease with which such African language passwords can be compromised.

1.2 Research statement

In response to the above, this research aims to investigate the security of passwords that are derived from African languages by assessing the quality of passwords that users are using in various South African industries.

The main question addressed in this research is whether an African language-based password is less likely to appear in a breached password list and therefore more difficult to crack. Researchers can of course use a brute force method to obtain the clear text password but this is less likely to be successful as some common African words are 12 to 16 characters long. For example, **babengamadoda**, which is a common Zulu word meaning

were men, is 13 characters long. According to normal password customs, this may be considered a weak password as it does not contain special characters and numbers but by adding two numbers and one special character, it becomes a strong 16 character password.

1.3 Research objectives

The first objective is to investigate what kind of passwords South Africans use on Microsoft Active Directory (AD) environments. This objective is important because many South African organisations employ some sort of AD environment to manage identities and passwords within the organisation. With Microsoft enjoying a 84.5% market share¹ one can assume that most organisations employ the Microsoft AD within their organisation. Moreover most organisations do not know what kind of passwords their employees are using. Not having an understanding of what kind of passwords the employees may be using may open the organisation to risk because the majority of passwords may be weak.

Passwords that are stored on the Microsoft AD environment may be affected by the organisation's password policy, which dictates how users choose a password to be used in the organisation and also in their private space.

The second objective is to investigate how many of these passwords appear in password lists on the Internet. National Institute of Standards and Technology (NIST) SP 800-63b 5.1.1.2² requires organisations to actively block passwords that are contained in breached password lists. Since actively blocking passwords may be hard for many organisations, as breaches and password lists are created on a regular basis, training users to think differently about passwords becomes paramount.

The third objective would be to create NTLM password hashes using African words and names and seeing how many of these passwords can be cracked using a Graphics Processing Unit (GPU) password cracking rig and also seeing how many passwords other researchers would be able to obtain from the same list.

1.4 Approach

The approach used in conducting this research can be summarised as:

¹<https://www.statista.com/topics/823/microsoft/>

²<https://pages.nist.gov/800-63-3/sp800-63b.html>

1. obtain password hashes currently being used in organisations,
2. crack these hashes,
3. analyse the cleartext passwords,
4. compare with previously breached password lists,
5. generate password hashes using African words and,
6. analyse the passwords cracked by independent researchers.

1.5 Limitations

The research question is based on African worded passwords, that is, how hard would it be for an attacker to obtain someone's password if it was written in an African language. As Africa has many indigenous languages the researcher is limited to the official South African indigenous languages. There are various writing systems around the world and most African languages follow the Latin alphabet writing system and thus, this research will focus on African words that are written in the Latin alphabet. Most wordlists downloaded from the Internet and other sources contain mainly English words/passwords and there are many English dictionaries that can be downloaded to create possible passwords. However, there are hardly any African language dictionaries and especially South African African language dictionaries.

- This research focuses on Microsoft Windows NTLM hashes that are obtained in South Africa.
- The research does not attempt to obtain all breached password lists.
- The research does not tackle the effectiveness of user awareness training on users and the types of passwords they select before or after such training.

1.6 Document structure

Chapter 2 of this research document contains a literature review of relevant literature.

Chapter 3 discusses the methodology of how the NTLM and LM hashes were obtained.

Chapter 4 analyses the cracked NTLM and LM hashes that were obtained in Chapter 3.

Chapter 5 presents a discussion of the results of organisational passwords that were successfully cracked.

Chapter 6 explains how passwords using various African languages were generated and analysed.

Chapter 7 summarises the research and the findings and presents some ideas for future research.

Chapter 2

Literature review

This chapter gives an overview of the creation and use of passwords, followed by a discussion of typical organisational policies and how these impact the choice of passwords. Also covered in this chapter is a detailed look at four South African organisational policies. The chapter further discusses the process of how the Microsoft AD database is structured and how network clients communicate with the database. Lastly this chapter discusses how Windows passwords are stored, used and attacked.

2.1 Overview of passwords

Passwords have become ubiquitous wherever authentication is required to gain access to a system. Organisations spend millions on their cyber security and according to Morgan (2015) the cyber security market reached \$75 billion in 2015 and is expected to reach \$170 billion by 2020. It has been stated that the cost of cybercrime in 2016 was over \$450 billion globally (Letsebe, 2017). According to Summers & Bosworth (2004) passwords are important as they are normally associated with a username, which is in turn linked to a person. Someone guessing the password could lead to that user being impersonated on that system and for the organisation can lead to potential risk and loss of information.

Passwords in the IT sector are still the dominant human computer authentication used as the primary access security even though there are many, including fingerprint authentication, biometric authentication and one-time passwords (Bonneau *et al.*, 2015). In the early years 1970s, organisational password were "*passwords were added to protect against practical jokes and researchers using more resources than authorized*" (Bonneau *et al.*, 2015).

2.2 Organisational password policy

Summers & Bosworth (2004) found that in one organisation they investigated, the company's policy stated that each account had to have a password of the user's initials. As a password policy is a public document within the organisation, any employee can read the policy and understand the requirements of the organisation for passwords. According to Summers & Bosworth (2004) a senior vice-president of the organisation had left the organisation and soon after they noticed that there were suspicious activities happening on one of their systems. When they conducted an investigation they found that the ex-employee's new company started producing similar products to that of the organisation. This happened because the ex-employee understood the company password policy and used it against the organisation to steal sensitive information as the ex-employee could guess the passwords that users would use on the critical system.

With the increased dependency on passwords as the main authentication method on most computer systems, it is imperative for the organisation to ensure that the passwords being used in the organisation are secure in transit, usage and storage. Password policies according to Inglesant & Sasse (2010) are there to preserve an organisation's confidentiality, integrity and availability. Poorly chosen passwords may open the organisation to unauthorised access of resources and possible financial losses. All users including contractors and vendors with access to the organisations systems and resources should comply and take appropriate steps in selecting and securing their passwords.

The password policy is there to explain the rules of how users should interact with computer systems in a consistent way. Moreover, when organisations create password policies it is to *"establish a standard for the creation of strong passwords, the protection of those passwords, and the frequency of change"* (SANS Institute, 2014).

According to Summers & Bosworth (2004) features that should be included in a good password policy are the following:

1. Passwords should:

- have a minimum length of six to ten characters
- include alphanumeric, upper-case, lower-case and special characters
- not consist of words that are found within a dictionary
- have digits appended to the end of the password

2. Users should:

- not reuse the previous five passwords
- not write any password down
- not share passwords
- change a password if it is suspected as having been compromised

3. Organisations should:

- disable a user after 30 days of inactivity or the user having left the company
- have a minimum password age of 10 days and a maximum of 60 days
- publish and educate users about the relevant password policy

The Center for Internet Security (CIS) has released benchmarks and best practice guidelines that organisation can freely download and implement on various platforms and computer systems within their organisations (Mavretich, 2012). The benefits of implementing the CIS benchmarks are that they are very granular and are broken into two levels. CIS level one recommendations are *"practical, provide a clear security benefit and do not inhibit the utility of the technology beyond acceptable means"* while level two recommendations are intended for environments where extreme security is paramount and for organisations that may want to add in defence in-depth measures (Center for Internet Security, 2016, p.24).

One of the CIS benchmarks is the CIS Microsoft Windows Server 2012 R2 benchmark v2.2.0¹. In this benchmark the CIS gives recommendations that organisations can follow when creating a password policy for the Microsoft AD environment. While the CIS benchmark is not a policy document that an organisation must implement, some of the sections contained within can be included in most organisations' password policies because of the wide use of Microsoft's AD.

The CIS benchmarks differ from the recommendations of Summers & Bosworth (2004) by stating that the organisation should implement a minimum password length of 14 characters. The CIS password length control may be in response to the ever increasing password cracking capabilities, which can achieve 350 billion guesses per second against Microsoft NTLM cryptographic algorithm (Goodin, 2012)². The CIS and Summers &

¹https://www.cisecurity.org/wp-content/uploads/2017/04/CIS_Microsoft_Windows_Server_2012_R2_Benchmark_v2.2.0.pdf

²[https://msdn.microsoft.com/en-us/library/windows/desktop/aa378749\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/aa378749(v=vs.85).aspx)

Bosworth (2004) however agree that the password age should be set to a maximum of 60 days and that passwords must meet certain complexity requirements.

The US based National Institute of Standards and Technology (NIST) released a draft specification for the 800-63-3 digital identity guidelines: authentication and lifecycle management (Grassi *et al.*, 2017). The NIST guideline differs from the CIS benchmark, as the former is geared towards general controls that the organisation can apply to multiple facets within the organisation whereas for each CIS benchmark the organisation must follow the guideline for that particular system.

The NIST guideline differs from both Summers & Bosworth (2004) and Center for Internet Security (2016) in the use, protection and creation of secure passwords. NIST is in favour of creating password policies that are more user friendly. According to Grassi *et al.* (2017) the NIST guideline puts the burden on the verifier (the organisation) which ensures that the security of the organisation resides with the organisation. This means that the organisation can stop asking users to improve their security because in reality some users do not improve the security of the organisation but actually put it at risk.

Grassi *et al.* (2017) differ from the CIS by stating that the minimum password length should be eight characters. While the CIS has been slowly increasing the password length over time to compensate for the increased power of password crackers, NIST has kept their recommendation at eight characters. One of the reasons for this may be that NIST is trying to shift the burden of security away from the user and to the organisation or the verifier. According to Grassi *et al.* (2017) the verifier should allow the user to create passwords that are:

- a maximum of 64 characters in length,
- include American Standard Code for Information Interchange (ASCII)³ characters as well as the space character,
- allow for the use of UNICODE⁴ characters in passwords including emojis.

In addition to the above controls, the NIST guideline differs from previous guidelines for creating password policies by stating that the organisation shall inspect the password against a list of known commonly-used, compromised passwords. This list may include

³<https://www.rfc-editor.org/rfc/rfc20.txt>

⁴http://standards.iso.org/ittf/PubliclyAvailableStandards/c063182_ISO_IEC_10646_2014.zip

"passwords obtained from previous breach corpuses, dictionary words, repetitive or sequential characters (e.g. 'aaaaaa', '1234abcd') and context-specific words, such as the name of the service, the username, and derivatives thereof" (Grassi *et al.*, 2017, p.14). If the password that the user wishes to use is included in this list, the verifier must reject that password and ask the user to create a new password.

This new way of checking passwords is different to what Summers & Bosworth (2004) and the CIS suggested as it is proactive in the creation of the password and ensures that the user creates a password that at least does not appear in a breached password list. It seems like the aim of the NIST 800-63-3 guideline is to put the burden of security on the verifier or the organisation but at the same time force users not to create passwords but instead create passphrases that are longer and thus harder to crack. The additional use of ASCII and UNICODE characters in passwords can help increase the number of characters that need to be checked to obtain the clear text password if the hash is known.

Another benefit of the NIST guideline is that the verifier *"should only expire a password if there is a good reason"* (Grassi *et al.*, 2017). This will help the user to create one strong password that they can remember instead of trying to create a new password every 60 or 90 days. Moreover NIST 800-63-3 is changing how two-factor authentication should be conducted, by stating that Short Message Service (SMS) should not be used because *"of the security of delivery of SMS, malware that can redirect text messages, mobile phone network attacks such as SS7 hack and mobile phone number portability"* (Gibbs, 2016, p.1).

The main issue with the NIST 800-63-3 guideline, is that it places a lot of emphasis on the organisation to improve the security posture; however, the security of the organisation is only as strong as its weakest link, mainly the employee.

If organisations implement the NIST 800-63-3 guidelines by actively blocking previously breached passwords and ensuring that users use complex passwords but the user uses the same password on a third party site and that third party gets compromised, it may put the organisation at risk without the organisation being aware of the third party breach affecting them.

2.2.1 Typical South African password policies

Four South African organisations' password policies are considered in this section to establish to what extent they conform to the guidelines discussed in the previous section.

The first password policy that is being reviewed belongs to Axiomatic Consulting⁵. This policy was selected because it is publicly available for anyone to download from their website. The second policy belongs to Cape Nature⁶. This policy was picked because it is publicly available from their website and the organisation included sections of the South African laws that affect the organisation.

The third organisational policy that was reviewed belongs to Mpowered⁷, a software solution company. This organisation was chosen because it is in the IT sector. The fourth organisation is Rhodes University⁸ and it was chosen because it had a breakdown for how passwords should be created for normal users and for servers.

The Axiomatic Consulting policy does not have version numbers or a creation date, which could mean that the password policy published online has not been updated or the organisation as a whole does not review the policy at least once a year. The organisation does not state how the password should be created but only that if the security of the password is in doubt, the user must change it. The organisation puts a lot of responsibility on the users to ensure that their passwords are kept secret as they will be responsible for all transactions that are conducted on their user accounts. This password policy lacks a lot of the basic sections such as password length, complexity, password history and user awareness.

The second organisation Cape Nature that published its *"Acceptable Use of Information Technology Information Systems"* included several sections that neither Grassi *et al.* (2017) nor Summers & Bosworth (2004) discussed. This policy has version numbers and who is responsible for the policies, and most importantly, is created around the various relevant South African National Acts (CapeNature, 2007):

- The Electronic Communications and Transactions Act 25 of 2002
- The Regulation of Interception of Communications Act 70 of 2002
- The Financial Intelligence Centre Act 11 of 2008
- Promotion of Access to Information Act 2 of 2000

⁵<https://www.axiomatic.co.za/wp-content/uploads/2016/08/Information-Technology-IT-End-User-Policy.pdf>

⁶<http://capenature.worldwidecreative.co.za/docs/418/CapeNature%20IT%20IS%20Policy%20April%202007.pdf>

⁷<https://www.mpowered.co.za/>

⁸<https://www.ru.ac.za>

- The Labour Relations Act 66 of 1995
- Basic Conditions of Employment Act 11 of 2002
- Copyright Act 98 of 1978
- Trade Marks Act 194 of 1993
- The Telecommunications Act 103 of 1996
- The Constitution of the Republic of South Africa Act 108 of 1996
- Common Law (data protection & privacy)
- Corporate Governance set out in King 2 of 2002

The organisation thus ensured that the password policy is created within the constraints of the laws of South Africa. This is important as the national laws of the country supercede the organisation's policies and regulations. The organisation does have some points that were mentioned by Summers & Bosworth (2004) such as keeping passwords secure, not sharing the same password between accounts and users being responsible for the security of their passwords and accounts.

When dealing with default passwords the second organisation makes it very clear that all new systems must be changed immediately at first use. None of the other password policies reviewed accounted for systems. The second organisation's password creation policy follows Microsoft's password policy⁹ and the criteria defined by Summers & Bosworth (2004) for creating a strong password. The second organisation includes rules that users must follow in creating a password, namely:

- the password must be at least seven characters long,
- the password must be alphanumeric,
- the password must contain a special character,
- the password may not be the same as the username,
- the password may not contain the word "*password*",
- the password may not be blank,

⁹[https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/hh994572\(v=ws.11\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/hh994572(v=ws.11))

- the password may not contain any substring with the same four letters used in the previous password,
- and passwords must be changed on a monthly basis.

Some of these rules appear in the NIST guidelines such as ensuring that the username is not used as a password and ensuring that sequences of four letters are not repeated in a new password. One aspect that is not discussed is whether the passwords being used are audited or verified before being accepted by the IT system.

The password policy gives suggestions on password choices for their users. As the password policy of the organisation requires users to change their passwords on a monthly basis, a list of example passwords is provided as shown in Figure 2.1. Giving users ex-

Password examples:

MONTH	Password	Description
January	JAN001#	7 long , one special character(Hash) , also capitalization
March	Ma03!*	7 long , 3 special characters + capitalization
August	Ugus657	7 long , start on second letter of the month + capitalization
February	BEF6612	7 long , first 3 letters reverse and capitalized.
February	Ef2323\$	7 long , first 2 letters reverse and capitalized with special character.
February	*feb002	7 long , one special character(Asterisk)
April	aPr4444	7 long , one capitalization(second letter of the month)
December	deC1212	7 long , one capitalization(third letter of the month)
September	\$sept06	7 long , one special character(Dollar) of the year 2006
October	OCT@1010	8 long , one special character(At) in the middle + capitalization

Figure 2.1: Password examples taken from CapeNature (2007)

amples of possible passwords in the password policy might mean that some users create passwords using the example passwords. CIS and NIST have indicated that passwords should be 8 characters and longer but in Figure 2.1 all the passwords except one is only 7 characters long. Summers & Bosworth (2004) give two examples of what constitutes a strong password: the sentence *"May the force be with you"* becomes *"Mt3%wU"* and *"I teach 3 classes at Columbus State University"* becomes *"It3c@cSu"*.

The policy of the third organisation (Mpowered, 2015) is a combination of the recommendations of Summers & Bosworth (2004) and the CIS. The organisation gives a guideline on what constitutes a strong password by stating that users must create a password that is 12 alphanumeric characters long and contains two upper-case characters, at least one special character and at least one number.

The organisation also gives examples of what constitutes a weak password: any password that is less than eight characters long, contains words that appear in a dictionary, foreign

language or exist in a language slang, dialect or jargon is a weak password. Moreover, employees of this organisation may not use their birthdates, addresses, phone numbers, names and/or names of family members and fantasy characters.

The organisation password policy also touches on a section that NIST covers in its 800-63-3 guideline, namely having passwords that contain patterns. If a password contains either a number pattern such as "123456" or a letter pattern such as "qwerty" it is a weak password. These are passwords that the NIST guideline aims to prevent. One of the reasons why the organisation has a password policy of 12 characters or more is because it is trying to move employees away from passwords and towards passphrases.

The fourth organisation is a South African university based in the Eastern Cape (University, 2016). The university follows Summers & Bosworth (2004) in how they created the password guideline such as creating a password that is eight alphanumeric characters long and contains special characters as well. Rhodes University, like the third organisation, makes it clear that no identifiable information should be used in passwords such as names (own and family members), pet names and birthdays. Furthermore, the password must not be a dictionary word, slang or jargon.

An interesting section in the password policy is the statement that the University carries out audits on all passwords and if the clear text passwords can be obtained from weak passwords, the user will be required to change that password. This checking of passwords by the University follows the NIST 800-63-3 control where the organisation must take a proactive stance in verifying that passwords are not weak.

The University also has two separate password policies for desktop administrators and server administrators, respectively. In addition to the general password guidelines as listed above, more controls are included for desktop and server passwords. Desktop and server passwords should be changed at least once every six months and all login attempts are logged and reviewed.

2.2.2 Password policy concerns

According to Komanduri *et al.* (2011, p.1) "*it is difficult to define the relationship between the components of a password-composition policy and the predictability of the resulting passwords*". This is supported by Blocki *et al.* (2013, p.1) who stated that "*a password composition policy restricts the space of allowable passwords to eliminate weak passwords*

that are vulnerable to statistical guessing attacks". When organisations create password policy guidelines they are based on theoretical estimates or small-scale laboratory studies and according to Komanduri *et al.* (2011, p.1) *"what makes designing an appropriate password-composition policy even trickier is that such policies affect not only the passwords users create, but also users' behaviour".*

The main issue with password policies and guidelines is that they give instructions to users on how they should create their passwords which could lead to users creating passwords that are either too difficult to remember causing them to write the password on paper or follow a similar pattern to their previous password. According to Summers & Bosworth (2004), for some organisations the enforcement of password policies had a detrimental impact on the organisation because it caused some users to have bad password management. As users were being forced to create complex passwords and to change them on a regular basis, these users might start taking short cuts in how they create their passwords such as adding numbers to the end of the password to increment them by one or add the current month as a number.

According to Parkin *et al.* (2016, p.23) as security is not foremost in a user's mind and is also not his/her primary task within the organisation, organisational password policies may negatively shape a user's behaviour towards creating passwords. The author found that when organisations devise password policies, the security controls described rarely consider the impact on the usability and productivity of the users.

The main issue for all organisations is that password policies can be circumvented by the user and the controls that have been put in place can be bypassed by the user in order for them to use their chosen password.

2.2.3 Selection of passwords

As discussed in Section 2.2.1, organisations can force users to create passwords according to their password composition policy (i.e. their password policy) to ensure that the passwords are harder to crack. According to Friendman (2014) 86.32% of South African users surveyed were confident that their personal passwords were strong and secure. However, the author found that only 52.14% indicated that credentials (username and password) were not adequate protection for their user account (Friendman, 2014). The majority (61 respondents) of those who indicated that usernames and passwords were not sufficient were in the IT industry.

Users develop a strategy when they have to create a password that must conform to the password policy constraints. According to Inglesant & Sasse (2010, p.6) when the security requirements or the password policy *"exceeds the users capabilities, they are forced to develop either more complex or, alternatively, less secure passwords"*. This is dependent on whether the system for which the user is trying to create the password, actively checks the password.

Inglesant & Sasse (2010) found that most users develop a strategy when creating password such as *"pairing words from a cycle of non-English words"* or recycling old passwords and making small changes.

A study conducted by Google with 2000 participants to understand the methods they used to choose their account passwords, found that people will use information that is around them or common to them when creating a password (Van Allen, 2013). This includes pet names, notable dates (for example wedding anniversary), family member's birthday, children's names, family member's names, their birthplace or that of someone in their family, favourite holiday, favourite sports team, name of a significant other and the word *"Password"*.

According to Dell'Amico *et al.* (2010) users most often knowingly create weak passwords because strong passwords would be harder for them to remember. The issue with creating weak passwords is that it may be easier for attackers to guess an easy password than a stronger password. This creates a trade-off between usability and security where the final decision made in creating the password falls on the user and as previously, before the user will often knowingly choose a weak password or try to circumvent security best practice.

2.3 Password attack methods

When performing offline attacks on user passwords, there are various techniques and methodologies that attackers can use in order to guess the user password. When attackers try to guess passwords, they are not testing the security of the system that the password is trying to protect, but rather trying to exploit the weakest link in any IT environment, that is, the end user. This section considers options available to attackers trying to crack passwords.

If an attacker has the hash of the password there are various methods that the attacker can use to obtain the clear-text password. Using freely available tools such as *hashcat*¹⁰

¹⁰<https://hashcat.net/hashcat/>

and *John the Ripper*¹¹ password cracking applications, an attacker can perform various attack methods on a password hash. According to Matt (2015) using either *hashcat* or *John the Ripper* the first method that the attacker would normally try is to run the password hash through a password dictionary and common passwords list. A password dictionary is a list of precompiled words that can be used to guess the password of the user (Whitaker & Newman, 2005).

2.3.1 Attacks using dictionaries

The mostly likely first method the attacker would try is to run a dictionary. Rankin (2012) states that an attacker uses a password dictionary to attack the user's password hash in the hope that the user is using a previously exposed password or some personal information such as pet names, family member or birthday, for example. A password dictionary may be tailor-made for the user targeted in the attack, but most of the time the password dictionary is a large collection of words and known weak passwords. The main advantage of using a password dictionary is that the attacker is restricted to the combinations contained within the dictionary list and therefore the amount of time needed to possibly obtain the clear-text password is reduced. The disadvantage of using a password dictionary is that the attacker is restricted to the combinations that are on the list and some passwords may be complex enough to not be in the list.

2.3.2 Brute-force attacks

According to Stuttard & Pinto (2011, p. 165) and Rankin (2012), a brute force attack is the next method that an attacker would use to guess a user's password. A brute force attack is an attempt to try every possible combination of letters, numbers and special characters to guess the password. In a brute force attack, attackers are only limited by the amount of time they are willing to spend on trying to guess the password. Guessing any password has a cost attached to it, which is normally the time an attacker is willing to wait to obtain the password.

¹¹<http://www.openwall.com/john/>

2.3.3 Attacks using rainbow tables

Using rainbow tables is a third attack method that an attacker could use to guess a password. Rainbow tables are computed hashes for each of the words in the dictionary or password dictionary that are stored in a hash table that the attacker could use to look up the hash value of the user password (Rankin, 2012). For a rainbow table to be useful, the computed hashes must have been obtained using the same hashing algorithm (for example Secure Hash Algorithm 1 (SHA1)) as that used for the password the attacker is trying to obtain.

Numerous authors have analysed the percentage of passwords that can be cracked as a function of the search space size. Dell'Amico *et al.* (2010) found that current researchers focus on passwords that have been discovered using a single attack method (dictionary attack for example) but do not quantify how strong the remaining passwords are against other forms of attack methods. To answer this question, the authors first studied other researchers and found that current researchers only provided a partial answer to the question of how many passwords could be guessed. They found that the percentage of broken passwords in a dictionary attack method varied between 17% to 24% dependent on the dataset and dictionary size. The authors added that by using their proposed technique that systematically mangled the words in dictionaries and datasets, they can discover up to a third of additional passwords.

2.3.4 Attacks using part-of-speech tagging

Ashwini *et al.* (2013) studied the use of long sentence-like passwords and how grammatical structures diminish the security of the passwords. Using parts-of-speech tagging which is "*a process of assigning a part of the speech to each word in a sentence*" (Ashwini *et al.*, 2013, p. 317) they found that if passwords follow English grammatical rules they could be attacked. In the English language there are eight parts of speech, namely nouns, verbs, adjectives, pronouns, adverbs, prepositions, conjunctions and interjections. They found that, since the use of sentence-like passwords was increasingly becoming the most popular way to create a password, researchers and organisations were recommending that users use longer but simpler compositional requirements in creating a password. The authors found that these passwords could be vulnerable to attacks. Typically, a minimum of 16 characters in a sentence or phrase-like password should be used, instead of complex smaller character passwords. The reason researchers and organisations prefer length over

complexity is that the length of the password increases the password entropy and according to Ashwini *et al.* (2013) makes the password more difficult to guess. As previously stated, creating a password is a compromise between security and usability and the user would most often create a password that is easier to remember.

According to Ashwini *et al.* (2013) the same can be said about users when creating sentence-like passwords where the user will create a longer password that conforms to English language grammar rules to help them to remember the password. The authors analysed 1434 passwords that consisted of 16 characters and found that 18% of the users chose passwords that contained grammatical structures. The passwords that were analysed were found to contain two or more sequential dictionary words that followed the English grammatical structure of *determiner adjective adjective noun* and other types of structures such as postal addresses, email addresses and uniform resource locators (URL).

2.3.5 Keyboard walking

Organisations are increasing password length to combat user passwords from being cracked easily. This action is supported by Center for Internet Security (2016) who stated that password lengths should be 14 characters and longer but the issue with increasing password length is that users become creative in the way they create passwords. Guidorizzi (2013) states that humans are not computers and therefore cannot remember random letters and numbers and therefore one technique that users use to compensate for this is to use *keyboard walking*. Keyboard walking is a technique that allows users to create passwords following a keyboard pattern. According to Guidorizzi (2013) *keyboard walking* is an overall security problem because it provides no challenge for the computer to discover the password created by the user even though the password may be compliant with various password complexity checks. Figures 2.2, 2.3 and 2.4 illustrate how *keyboard walking* can be used to create passwords which may seem complex but can be easily guessed. Figure 2.2 shows examples of possible passwords which can range from 8 up to 12 characters. While this is not an exhaustive list of possible combinations, we can see that either way the passwords follow a pattern.

Steube (2018) stated that when users employ *keyboard walking* they typically start on the left hand side of the keyboard; thus letters like '1' and 'q' are well known starting points

¹²https://bytesdarkly.com/images/2014/08/kb_walk1.png

¹³https://bytesdarkly.com/images/2014/08/kb_walk21.png

¹⁴https://bytesdarkly.com/images/2014/08/kb_walk3.png

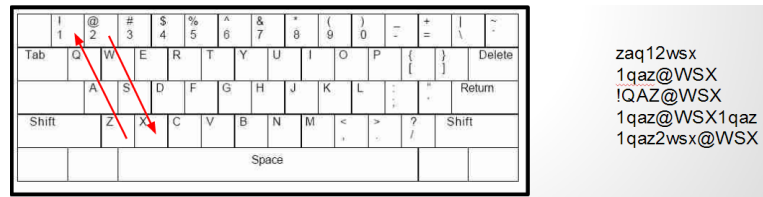


Figure 2.2: Example of keyboard walking¹²

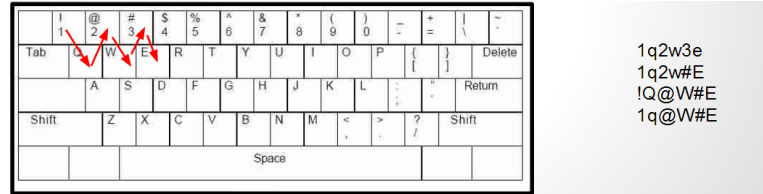


Figure 2.3: Second example of keyboard walking¹³

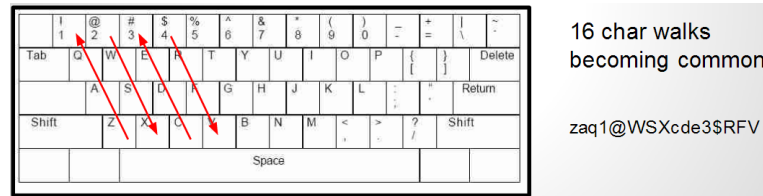


Figure 2.4: Third example of keyboard walking¹⁴

as shown in Figures 2.2, 2.3 and 2.4. The middle of the keyboard is interesting because *keyboard walking* can take the shape of a square and a password like *rtynhbuf* or *vbnhytrf* which may seem random, is still considered a weak password as illustrated in Figure 2.5.

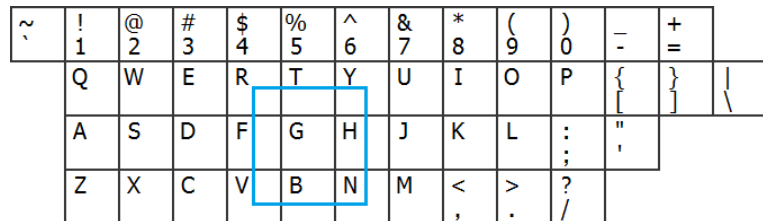


Figure 2.5: Square keyboard walking¹⁵

According to Steube (2018) *"there are only 9 possible geographic directions"* on a keyboard based on the design. These directions are:

- south-west
- south
- south-east

¹⁵https://github.com/hashcat/kwprocessor/blob/master/doc/img/en_square.png

- west
- stay
- east
- north-west
- north
- north-east

Users can use directions to create passwords such as *qwerfdsa* which follows *START* -> (*base-character = q*) + 3 * *EAST* + 1 * *SOUTH* -> 3 * *WEST*. Figure 2.6 illustrates this. Figure 2.7 shows a rectangular east to west pattern where the password pattern

~	!	@	#	\$	%	^	&	*	()	-	+	
	1	2	3	4	5	6	7	8	9	0	{	}	\
	Q	W	E	R	T	Y	U	I	O	P	{	}	\
	A	S	D	F	G	H	J	K	L	:	"	'	
	Z	X	C	V	B	N	M	<	>	?			
								,	.	/			

Figure 2.6: Directional keyboard walking¹⁶

doubles up on the bottom three characters. This password *rfvbnvbv* can still be easily attacked as the pattern can be guessed.

~	!	@	#	\$	%	^	&	*	()	-	+	
	1	2	3	4	5	6	7	8	9	0	{	}	\
	Q	W	E	R	T	Y	U	I	O	P	{	}	\
	A	S	D	F	G	H	J	K	L	:	"	'	
	Z	X	C	V	B	N	M	<	>	?			
								,	.	/			

Figure 2.7: Rectangular east and west pattern¹⁷

2.4 Microsoft active directory database

In a Microsoft AD environment, all directory data is stored in the *NTDS.dit* Extensible Storage Engine (ESE) JetBlue database (Microsoft, 2014). NTDS stands for NT

¹⁶https://raw.githubusercontent.com/hashcat/kwprocessor/master/doc/img/en_rect1.png

¹⁷<https://github.com/hashcat/kwprocessor/blob/master/doc/img/enexrfvbnvbv.png>

Directory Services and the DIT extension stands for Directory Information Tree (Rahman, 2014). Figure 2.8 shows the Active Directory data store files, consisting of the

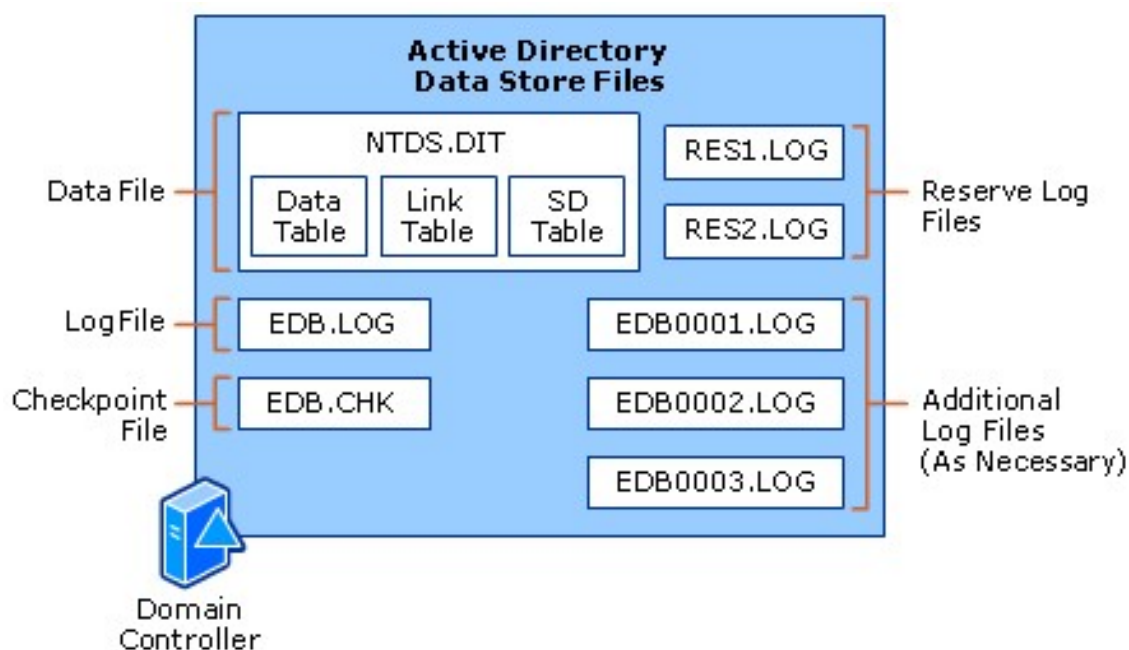


Figure 2.8: Physical components of the Data store taken from Microsoft (2014)

NTDS.dit, *EDB.LOG*, *EDB.CHK*, *RES1.LOG* and *RES2.LOG*. The *NTDS.dit* data file contains the Data table, Link table and Security Descriptor (SD) table. *NTDS.dit* is stored in two default locations on a domain controller, *systemroot\NTDS\NTDS.dit* and *systemroot\System32\NTDS.dit* (Microsoft, 2014). The two copies of the *NTDS.dit* file each have different functionality on the domain controller. The first copy is stored in the NTDS folder which stores the database that is in use on the domain controller, while the System32 *NTDS.dit* copy is a distributed copy that is used when installing AD on a server running Windows Server 2003 or later.

The Data table, contained within the *NTDS.dit* file, is important as it contains all the AD information such as users, groups, password hashes, and other AD data (Microsoft, 2014). Data in the Link table defines linked attributes whose values refer to other AD objects. For example the Link table is used to link attributes contained within the Data table such as **MemberOf** attribute, which links user objects that are contained within other groups that belong to the user.

2.4.1 Password storage

The Lightweight Directory Access Protocol (LDAP)(RFC4511) is *"a directory service protocol that runs on a layer above the TCP/IP stack on port 389"* (Microsoft, 2018a). It provides a mechanism for network devices or users to connect to the LDAP service and be able to search and modify attributes. AD gives authenticated users the ability to change their password. Microsoft Windows systems change user passwords by performing an LDAP modifying operation (Microsoft, 2018b). Figure 2.9 shows how network clients interact with the domain controller via LDAP.

LDAP provides the mechanism for network clients to interact with the Directory System Agent (DSA). The DSA runs as the *Ntdsa.dll* on the domain controller and provides

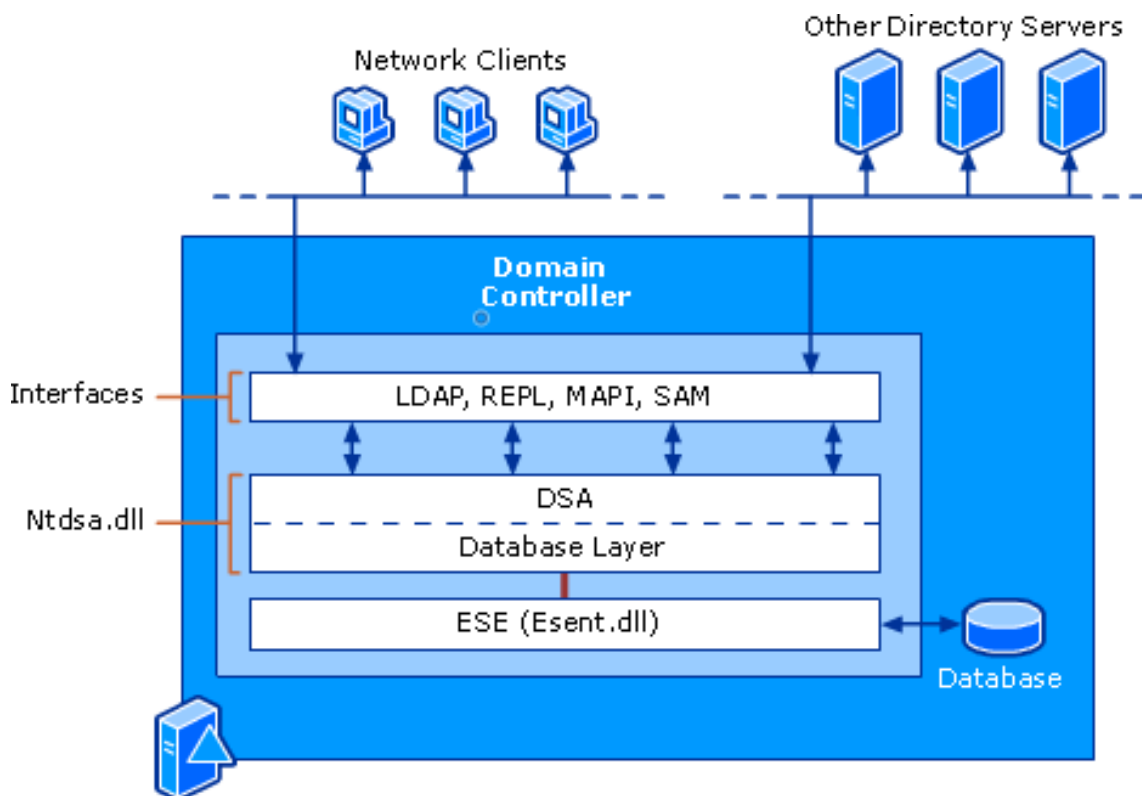


Figure 2.9: Data store architecture taken from Microsoft (2014)

the means for network clients to gain access to the directory database. *Ntdsa.dll* provides an Application Programming Interface (API) between applications and the directory database to *"protect the database from direct interaction with applications"* (Microsoft, 2014). The ESE manages the tables and all the data is stored inside the database file.

Microsoft AD *"stores the password on a user object or inetOrgPerson object in the unicodePwd attribute"* (Microsoft, 2018c). The network client uses the LDAP connection to

modify the attribute by following the process illustrated in Figure 2.8. The connection process is different depending on the Microsoft Windows server operating system installed.

Microsoft Windows Server 2000 operating system requires that the network clients have a 128-bit Secure Sockets Layer (SSL)/ Transport Layer Security (TLS) encryption to modify attributes on the domain controller. On Microsoft Windows Server 2003 operating systems and later, network clients can modify the *unicodePwd* attribute provided that they use 128-bit Simple Authentication and Security Layer (SASL)¹⁸ encryption.

When a domain controller receives an LDAP modification request from a network client, the domain controller first checks if the request contains a delete (*Vdel*) operation followed by an add operation (*Vadd*) for the *unicodePwd* attribute (Microsoft, 2018c). If the attribute is being modified the domain controller considers the request to be a password change. For the password change to be successful the domain controller must ensure that the user is allowed to change their password and that the *Vdel* value contains the correct current password. However, if the password change is being conducted by an administrator, the domain controller responds differently to when a user conducts a password change. If the domain controller receives a single operational value of *Vrep* for the *unicodePwd* attribute, the domain controller considers the *"the request to be an administrative password reset and modify the password without knowledge of the old password"* (Microsoft, 2018c).

2.4.2 Windows password hashing algorithms

Older Microsoft systems such as Microsoft Windows NT 4.0, Windows 2000, Windows XP and Windows 2003 never stored user passwords in clear-text but used one of two algorithms LM and NTLM. NTLM hashes, which are also called UTF-16 encoded Unicode string or MD4 hashes, are used for authentication on modern Microsoft Windows operating systems and are stored in the Microsoft AD database. According to Microsoft (n.d.b) and Johansson (n.d.), technically the LM algorithm is not a hashing algorithm and is also less secure than NTLM. However, the reason that LM is still in use is for backward compatibility with older applications and operating systems such as Microsoft Windows 98.

According to Microsoft (n.d.b) and Johansson (n.d.), when a Microsoft Windows operating system is required to use the LM algorithm, it converts all the lowercase characters in the password to uppercase. If the password is shorter than 14 characters, the password is

¹⁸<https://www.ietf.org/rfc/rfc2222.txt>

padding with NULL characters until it is exactly 14 characters long and importantly LM does not support passwords longer than 14 characters. The password is then split into two 7-character parts as illustrated in Figure 2.10 and each part is encrypted with a Data Encryption Standard (DES) key that is 64 bits long¹⁹ according to (National Institute of Standards and Technology, 1999). The two cipher texts are concatenated into a 128 bit LM hash.

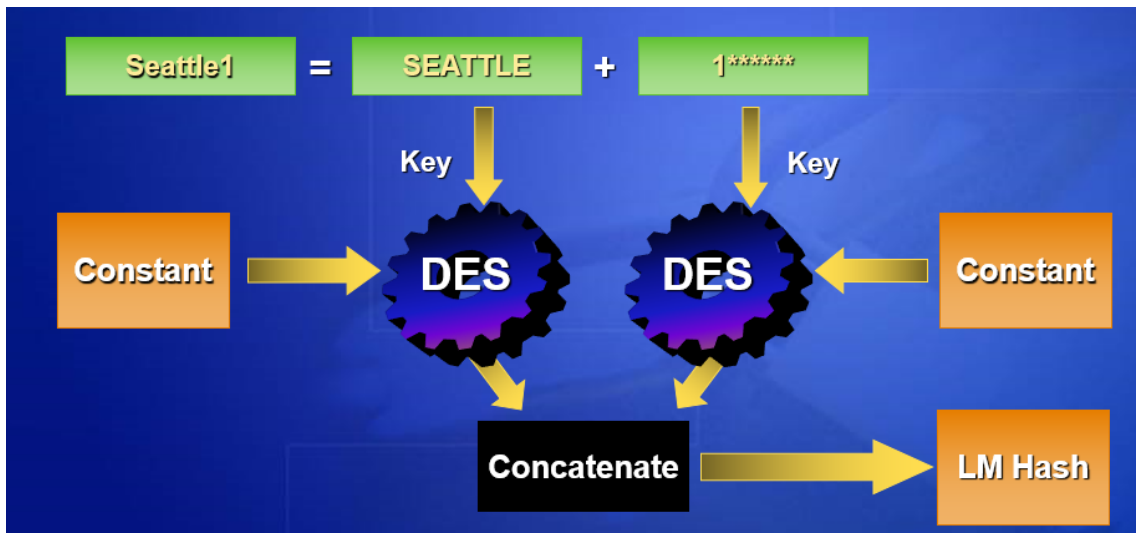


Figure 2.10: LM hash creation from Johansson (n.d.)

The LM algorithm has several weaknesses (Johansson, n.d.; Microsoft, n.d.b). The first weakness is that it accepts only common alphanumeric sets and does not accept unicode characters. LM is also case insensitive as it capitalises the password before applying DES encryption. Furthermore, since LM accepts 142 ascii symbols only 69 are available on an English keyboard (26 letters, 10 numbers, 33 punctuation), this means that the maximum number of passwords is 69×10^{14} where 69 is the English characters and 10^{14} is the maximum length of the LM hashing.

Figure 2.11 illustrates how an NTLM hash is generated. The NTLM hash is created by taking the plain text password, applying the Message Digest 4 (MD4) algorithm and storing the unicode hash in the Microsoft AD database. According to Johansson (n.d.) NTLM differs from LM hashing by being case sensitive, allowing 65,535 symbols unlike LM's 142 symbols and having a maximum length of 127 characters.

¹⁹<https://www.britannica.com/topic/Data-Encryption-Standard>



Figure 2.11: NTLM hash creation from Johansson (n.d.)

2.5 Analysis of Chinese regional breached passwords

Li *et al.* (2014) analysed over 100 million leaked and publicly available passwords from popular Chinese websites and compared them to English based leaked passwords. Table 2.1 shows the datasets that were analysed by Li *et al.* (2014). The authors assumed that all the passwords obtained from the Chinese websites only contained *Chinese user* passwords because the websites only provided Chinese web pages.

Table 2.1: Li *et al.* (2014) leaked password datasets

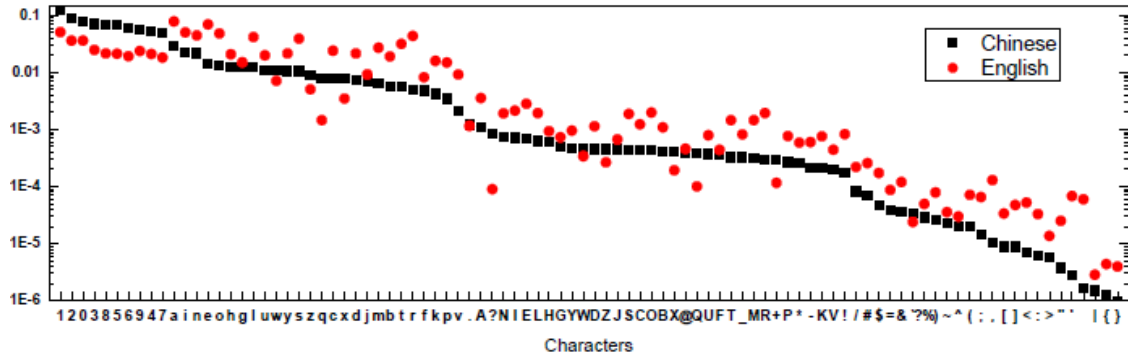
Dataset	Language	Distinct Accounts
CSDN - http://www.csdn.net/	Chinese	6,423,483
Tianya - http://www.tianya.cn/	Chinese	26,223,020
Duduniu - http://www.duduniu.cn/	Chinese	15,131,833
7k7k - http://www.7k7k.com/	Chinese	5,940,099
178.com - http://www.178.com/	Chinese	9,072,804
RockYou - http://www.rockyou.com/	English	32,602,882
Yahoo - http://yahoo.com	English	442,837

Li *et al.* (2014) found that Chinese users preferred to use digits in their passwords while English users preferred letters especially lower-case letters; however *"the password strength against guessing is similar for both groups as they share similar security concerns"*. One area in which the Chinese and English passwords differed was around dates. Li *et al.* (2014) found that *"Chinese users preferred dates with the year at the beginning while English users preferred it at the end"*.

Table 2.2 shows the top weak Chinese and English passwords. Analysing the two groups of weak passwords there are many similarities with users preferring to use digits in passwords. This contradicts the earlier finding of Li *et al.* (2014) that English users preferred to have letter-based passwords since Table 2.2 shows that the top three passwords are all

Table 2.2: Top weak Chinese and English passwords obtained from Li *et al.* (2014)

Chinese	English
123456 (2.17%)	123456 (0.88%)
123456789 (0.65%)	12345 (0.24%)
111111 (0.59%)	123456789 (0.23%)
12345678 (0.39%)	password (0.18%)
000000 (0.34%)	iloveyou (0.15%)

Figure 2.12: Top weak Chinese and English characters obtained from Li *et al.* (2014)

numbers. Figure 2.12 shows the frequency of each of the characters as analysed by Li *et al.* (2014). The authors checked all the characters (a-z, A-Z), digits (0-9) and all printable special characters. The characters were ranked in descending order according to the Chinese characters and the English characters ranked against the Chinese characters. The top ten characters in Chinese are the number characters from 0 to 9, while in English, character *a* has the highest frequency. Symbols or special characters are the lesser used characters in both Chinese and English. The special character *'* is the most used special character while special characters *'{'* and *'}'* are the least used characters. According to Li *et al.* (2014) special character *'?'* is the most frequently used character in Chinese passwords.

Keyboard walking has been discussed in Section 2.3.5. Li *et al.* (2014) found that both Chinese and English users employed keyboard patterns to enable them to remember their passwords. The authors analysed the user created passwords and found that 8.31% of Chinese and 2.42% of English based passwords used the same row keyboard pattern. The same row keyboard pattern is when users use consecutive sequences of characters in the same row, e.g. *qwerty*. The authors also found that 0.26% and 0.06% of Chinese and English users, respectively, employed the Zig Zag keyboard pattern, where sequential characters that are adjacent to each other are used, for example, *qawsxd*. Lastly the snake keyboard pattern is a sequence of characters where the keys are adjacent to each other on the keyboard, such as *asdfvbnm*. The authors found that 0.27% and 0.08% of Chinese

Table 2.3: Top five Chinese Pinyins and English words obtained from Li *et al.* (2014)

Top Chinese Pinyins	Top English Words
woaini (1.47%)	password (1.28%)
li (1.06%)	iloveyou (0.98%)
wang (0.97%)	love (0.76%)
tianya (0.89%)	angel (0.59%)
zhang (0.84%)	monkey (0.45%)

and English users, respectively used this keyboard pattern.

According to Li *et al.* (2014) Chinese Pinyins were developed in the 1950s and are "*designed to represent the pronunciation of Chinese characters*". Even though there are a few Chinese dialects the Pinyin characters are the same. As the Chinese users use the same keyboard as English users (Li *et al.*, 2014), people are trained from a young age to use Pinyins. These "*Pinyins have become the popular method to input Chinese characters to a computer*" (Li *et al.*, 2014) because they require no additional training. Since Chinese websites do not support passwords that are composed of Chinese characters directly, this only adds to Pinyins being the most popular method of creating passwords.

There are about 420 Pinyins and when Chinese users create passwords using Pinyins they usually combine multiple Pinyins to generate a password for example *nihao* is composed of Pinyins *ni* and *hao* (Li *et al.*, 2014). English based users employ base words which are words that cannot be broken down any further from which to create passwords, for example, *password*. Table 2.3 shows the top five Pinyins and English words that were found by the authors in their research.

Numbers are important in the creation of passwords. Representing dates as numbers is a method that is used to remember the numbers included in passwords. There are different methods that a user can employ when formatting dates. Li *et al.* (2014) condensed six-digit date formats into three categories:

- YYMMDD - Year, Month, Day
- MMDDYY - Month, Day, Year
- DDMMYY - Day, Month, Year

Eight-digit dates comprised the following three date formats:

- YYYYMMDD - Year, Month, Day

Table 2.4: Eight character date formats used obtained from Li *et al.* (2014)

	Consecutive Exactly Eight Digits	YYMMDD	MMDDYYYY	DDMMYYYY
CSDN	1,621,954	29.24%	0.25%	0.43%
Tianya	3,639,517	36.26%	0.35%	0.60%
Duduniu	1,700,329	28.87%	0.28%	0.84%
7k7k	2,470,204	32.41%	0.18%	0.37%
178.com	995,832	30.46%	0.13%	0.19%
RockYou	929,987	2.64%	7.70%	17.66%
Yahoo	6,981	2.78%	12.00%	11.17%

Table 2.5: Six character dates used obtained from Li *et al.* (2014)

	Consecutive Exactly Eight Digits	YYMMDD	MMDDYYYY	DDMMYYYY
CSDN	809,050	27.21%	4.04%	1.24%
Tianya	9,477,069	23.93%	3.05%	1.19%
Duduniu	2,688,347	17.84%	2.97%	1.78%
7k7k	3,999,958	24.34%	2.63%	0.88%
178.com	2,525,254	13.96%	1.72%	1.30%
RockYou	2,758,871	5.63%	21.90%	18.42%
Yahoo	21,020	4.66%	25.99%	7.77%

- MMDDYYYY - Month, Day, Year
- DDMMYYYY - Day, Month, Year

Table 2.4 shows the types and frequency of dates exactly eight digits long that were used in the various leaked password lists. Table 2.5 shows the six-digit dates that were found in the various leaked password lists and their frequencies. The positioning of dates was also important to Li *et al.* (2014) as they analysed passwords that contained letters and dates as passwords but eliminated digit only passwords. The authors found that for both Chinese and English users having dates at the end of the password was preferred and these were rarely placed in the middle of the password.

2.6 Summary

This chapter provided a brief history of passwords and how passwords were abused in the early years of access control. We looked at how the various international organisations such as NIST, Microsoft and CIS create benchmarks and guidelines that cover many aspects of cyber security as well as passwords within the organisation. NIST and CIS password guidelines shape how organisations in South Africa create password policies, which in turn shape how users create passwords.

Most organisations have password policies that dictate to the employees and third party vendors how they should create, use and remove passwords within the organisation. Some password policies are rigid and give clear instructions to the user about password security while other organisations are more flexible in their rules. We examined four South African organisations' password policies and found that their password policies followed international guidelines from Microsoft and CIS. Users develop a strategy, for creating passwords, which if it works in the organisation, is reused in other systems. There are various attack methods that can be used to attack password hashes ranging from brute force attacks, dictionary attacks to using parts-of-speech tagging to attack longer passwords or passphrases.

This chapter explained how passwords are stored in Microsoft AD environments and how network clients can initiate a password change. Also discussed were the two hashing algorithms NTLM and LM, and how each hashing algorithm creates the hash before storing it in the NTDS.dit database file. LM hashing algorithm weaknesses and how LM can be attacked easily were discussed with caution that LM hashing should not be used for password storage.

Chapter 3

Research design

This chapter gives an overview of the research undertaken and the steps applied to analyse the passwords. It also discusses the password lists and rules used to crack the password databases available.

3.1 High-level overview of research

One of the objectives of this research is to analyse a sample of passwords that users are using within corporate environments in South Africa. To understand these, the first step is to obtain the LM and NTLM hashes. A brute force algorithm is used against the LM hash dataset because of the inherent weakness of the hashing algorithm, while NTLM hashes are attacked using breached password lists and rules. A secondary objective is to gauge whether using African words as passwords can increase the security of the passwords. To achieve both these objectives the research process includes the following steps.

Step 1: Extract LM and NTLM hashes from password databases obtained from corporate organisation as discussed in the next section.

Step 2: Remove data that is not required for the research such as computer account hashes and blank account hashes as they do not contain any passwords.

Step 3: Analyse the extracted LM and NTLM hashes.

Step 4: Brute force the LM hashes that were collected from the corporate organisations password databases.

Step 5: Use breached password lists and rules to crack NTLM hashes that were collected from corporate organisations password databases to obtain clear text passwords.

Step 6: Analyse clear text passwords obtained and user password creation behaviour, e.g use of numbers, characters and special symbols.

Step 7: Create passwords in the African languages using the patterns observed in the previous step.

Step 8: Analyse the results of cracked African passwords.

These steps are discussed in detail in Chapters 4 to 6.

3.2 Password databases

The researcher had written consent from a registered cyber security company in South Africa to use various NTLM and LM hashes available from the company. These hashes had been collected by the cyber security company in the course of its business from various organisations in South Africa. The researcher is a subcontractor of the cyber security company from which the password hashes were obtained. Moreover, all LM and NTLM hashes used in this thesis and especially for the password cracking process were cleaned to remove any identifiable information such as account name, company name, and user first names and surnames. As further protection, all the password hashes and the cleartext passwords that were obtained for the purpose of this research were correctly destroyed after testing.

No ethical approval was thus required for this research, as the researcher was not involved in the collection of the password hashes.

Hashes from twelve different organisations were selected from those made available by the cyber security company such that each organisation comes from a different sector of the industry in South Africa. Different sectors of the industry were chosen so as to get a general understanding of the passwords employees use in various sectors of the economy. The password databases used consisted of LM and NTLM hashes extracted from organisations in the mining, financial, information technology, medical, government, services and manufacturing sectors.

3.3 Breached password lists

Three password lists were used against the NTLM password hashes to obtain the clear text passwords. The three password lists are discussed below:

3.3.1 HaveIBeenPwned

The first breach password list that was used is the *HaveIBeenPwned* password list downloaded from hashes.org. The *HaveIBeenPwned* password list consists of 517 million passwords.

3.3.2 Breachcompilation

The *breachcompilation* is a password list that contains 1.4 billion passwords. The password list torrent magnet link was obtained from <https://gist.github.com/scottlinux>. The file was not checked for duplicates and was used as is in the research as the *breachcompilation* password list is a collection of many password lists combined into one.

3.3.3 Weakpass_2a

The *weakpass_2a* password list consists of 7.88 billion passwords. According to weakpass.com¹ the password length ranges from 4 to 25 characters. Clear text passwords with 10 characters have the highest count in this password list followed by passwords that consist of 9 characters. Password masks give an indication of the format of the password. In the *weakpass_2a* password list the following are the top three masks with the number of occurrences given in brackets:

1. ?l?l?l?l?l?l (308,915,776 occurrences)
2. ?l?l?l?l?l?l?l?l?l (133,701,766 occurrences)
3. ?l?l?l?l?l?l?d?d?d?d (122,007,905 occurrences)

¹<https://weakpass.com/wordlist/1861>

The listed masks above show that the top password mask in the *weakpass_2a* list consists of all letters, represented by *?l*, and is up to 6 characters in length. The second top password mask contains only letters and is up to 8 characters in length while the third most frequent password mask consists of 6 characters followed by 4 digits.

3.4 Rules used

Hashcat allows for rule-based attacks. According to hashcat (n.d.) "*rule-based attack are one of the most complicated of all the attack modes*" that are supported by hashcat. Rule-based attacks are powerful because they can be designed like a programming language where words can be modified, cut, extended and given other conditional operations to make them flexible and effective at attacking. The rules that were used for the research are:

- InsidePro-HashManger.rule
- dive.rule
- `_NSAKEY.v2.dive.rule`

These rules were selected because they cover all of the tasks that the researcher would have wanted to be applied to the passwords. The *NSAKEY.v2.dive.rule* has over 123289² rules built-in by the author, which is sufficient for this research. An example rule would be to uppercase the first character and add a random number at the end. Another example rule would be to concatenate two words from the breached password list into one word to test whether that combination of words exists as a password.

3.5 Creating African language derived passwords

Nine Southern African dictionaries were obtained for this research. The following dictionaries, are licensed according to the GNU General Public License, were obtained from Kokuaviewer.org³:

²<https://github.com/NSAKEY/nsa-rules>

³<http://app.kokuaviewer.org/dics/>

1. South Ndebele - nr_ZA
2. Sepedi (Northern Sotho) - ns_ZA
3. Swati - ss_ZA
4. Southern Sotho - st_ZA
5. Tswana - tn_ZA
6. Tsonga - ts_ZA
7. Venda - ve_ZA
8. Xhosa - xh_ZA
9. Zulu - zu_ZA

Some of the words contained in the nine dictionaries were used to create African language passwords. The words that would be used for these passwords were randomly selected but were only included if they were six characters or longer. This was done to create semi-realistic passwords that conform to the average South African password policies.

The analysis of the organisational passwords would provide initial insight into how users create passwords within corporate South Africa. The selected African words were used to derive passwords that follow the same user password creation process. For example, if a user created the password *Hello1*, the equivalent African password could be *Sawubona1*. The length may differ but the capitalisation, number and special character placements are similar to how users create English passwords.

A total of 198 African words were used and all the words selected were the same for each African language. For example, *Hello* is the English word, *Sawubona* is the Zulu word, and *Dumelang* is the Northern Sotho equivalent. This was done to test whether some languages are easier to crack than others.

Once the African passwords had been created, they were hashed using the NTLM hashing algorithm and MD5. The NTLM hashing algorithm was used because the password is hashed as is, whereas the LM hashing algorithm uppercases all the characters and limits the password to 14 characters before hashing.

The hashed African passwords were subsequently sent to various password cracking forums and websites to test how effective they were against these services.

3.6 Hardware/software configuration

The hardware used is important when cracking passwords. Cracking NTLM and LM hashes requires GPU⁴ power more than Central Processing Unit (CPU). Prior to cracking the hashes the researcher set up the environment to obtain clear text passwords. The hardware and software used in the research are shown in Tables 3.1 and 3.2, respectively.

Table 3.1: Hardware used to extract and analyse the NTLM and LM hashes

Hardware	
CPU	Intel i7-8700k
RAM	16GB
GPU	2x Nvidia GeForce GTX 980
HDD	120GB SSD
HDD	5TB Hard Drive

Table 3.2: Software used to obtain clear text passwords

Software	
OS	Microsoft Windows 10 1803
GPU Drivers	Nvidia GeForce Driver 398.11
Hashcat	Version 4.1.0

3.7 Hashcat attack modes

Hashcat⁵ is the tool that was used to obtain the clear text password from the NTLM hashes available. Hashcat is a password recovery tool that requires GPU power to crack passwords and can be run on either Linux, MacOS or Microsoft Windows operating system. Hashcat supports brute force attacks and dictionary attacks against NTLM and LM hashes.

In addition to the hardware and software the researcher had to obtain wordlists for the dictionary attack on the NTLM and LM hashes. A wordlist is a text file that contains a collection of words that can be used in a dictionary attack.

⁴<https://www.infosecurity-magazine.com/news/gpu-cluster-can-crack-any-ntlm-8-character-hashed/>
<https://arstechnica.com/information-technology/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours/>

⁵<https://hashcat.net/hashcat/>

As stated in Section 3.6, hashcat was used to crack the LM and NTLM hashes. According to hashcat.net⁶ hashcat is the world's fastest password cracker and is the world's first and only in-kernel rule engine application.

Hashcat supports multiple attack modes such as dictionary attacks in which it tries all words in a file to obtain the clear text password. For combinator attacks, hashcat concatenates words from multiple wordlists to create a unique word to test against a hash. Another attack mode that is supported in hashcat is brute-force attack where the application tries all the characters on a keyboard to guess the password.

Hashcat supports a more effective way of brute forcing called mask attacking. In mask attacks hashcat tries all the combinations in a given key space by applying logic to the guess. Hashcat also supports a hybrid attack, where the application uses a combination of wordlists and mask attack. The last form of attack is a rule-based attack where hashcat applies rules to words that are contained within a wordlist.

3.8 Summary

This chapter explained the process for attacking the LM and NTLM hashes and what systems were used to perform the attack. The various rules and password lists that were used to obtain the clear-text password were also mentioned in the chapter. Deriving African passwords was discussed including how the passwords were created after analysing the clear text passwords.

⁶<https://hashcat.net/hashcat/>

Chapter 4

Analysis of LM and NTLM hashes

This chapter covers the NTLM and LM hashes used in this research, including the various applications and scripts that assisted in obtaining the clear text passwords and analysing the NTLM and LM hashes. Also discussed is the process of extracting NTLM hashes from the organisation's NTDS.dit file and the approach used to obtain the clear text passwords.

4.1 Extraction software

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 ::
```

Figure 4.1: Extract of Impacket's secretsdump

In Sections 2.4.2 NTLM and LM hashing algorithms were introduced. Once the hashes have been extracted from the NTDS.dit database, various tasks take place before they are compared and cracked. The NTLM and LM hashes were extracted from the NTDS.dit database using *impacket's secretsdump* Python script as shown in Listing 4.1. Figure 4.1 is an extract from *impacket's secretsdump.py* illustrating how the LM and NTLM hashes are displayed along with additional information. The username of the LM and NTLM hashes is the first to be displayed followed by the account Security ID (SID). The account SID according to Savill (2000) is a unique identifier for Microsoft objects such as users or groups, with SID 500 used for a domain administrator account.

After the SID, the first hash displayed is the LM hash, as described in detail in Chapter 2 and illustrated in Figure 2.11. The LM hashes have a major disadvantage that

organisations should be aware of to prevent them from being used for password hashing and storage. The hash that follows the LM hash is the NTLM hash which has also been described in detail in Chapter 2 and illustrated in Figure 2.11.

There are multiple ways that one can extract NTLM and LM hashes from the NTDS.dit database. There are built-in Microsoft Windows applications and Microsoft PowerShell libraries that can assist in extracting the NTDS.dit and SYSTEM file from Microsoft AD environments. The Microsoft utilities include:

1. *Vssadmin*, which makes a volume shadow copy and allows extraction using the shadow copy¹,
2. *NTDSUtil*, which is part of the Microsoft AD diagnostic tools and allows administrators to perform database maintenance²,
3. and leveraging snapshots of domain controllers that have been virtualised.

There are also many third-party tools that can interact with the Microsoft AD database such as the PowerSploit³ post-exploitation framework PowerShell, CrackMapExec⁴, which uses Benjamin Delpy's *mimikatz*⁵ and *ntdsxtract*, which is an AD forensic framework⁶.

Using built-in Microsoft Windows AD *NTDSUtil* one can extract the NTDS.dit, *SYSTEM* and *SECURITY* files from a running domain controller environment. Figure 4.2 shows the extraction commands and process to extract a copy of the NTDS.dit file. To extract the NTDS.dit file there must be a connection to a domain controller and the user must have administrative privileges on the system. Once the *NTDSUtil* is running, NTDS must be set to active. Once the NTDS is the active instance within *NTDSUtil* the next command that must be run is *ifm*, which allows the user to write the NTDS.dit file to disk and make the domain controller read-only (Microsoft, 2016). The *NTDSUtil* creates a snapshot, extracts the files that are needed, and stores them in the user-defined folder.

Figure 4.3 illustrates the process using *vssadmin* for the NTDS.dit extraction. The *vssadmin* application first creates a volume shadow copy of the drive containing the NTDS.dit

¹<https://technet.microsoft.com/en-us/library/dd348398.aspx>

²[https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/cc753343\(v=ws.11\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/cc753343(v=ws.11))

³<https://github.com/PowerShellMafia/PowerSploit>

⁴<https://github.com/byt3bl33d3r/CrackMapExec>

⁵<https://github.com/gentilkiwi/mimikatz>

⁶<https://github.com/csababarta/ntdsxtract>

```

Administrator: Command Prompt - ntdsutil.exe
C:\Windows\system32>ntdsutil.exe
ntdsutil.exe: activate instance ntds
Active instance set to "ntds".
ntdsutil.exe: ifm
ifm: create full C:\extracted
Creating snapshot...
Snapshot set {c26438d8-f25e-47ab-9016-2300a6768083} generated successfully.
Snapshot {2d42528d-9ce6-43a6-a16a-95a77d70f966} mounted as C:\$SNAP_201804221420
_UOLUMECS\
Snapshot {2d42528d-9ce6-43a6-a16a-95a77d70f966} is already mounted.
Initiating DEFRAGMENTATION mode...
Source Database: C:\$SNAP_201804221420_UOLUMECS\Windows\NTDS\ntds.dit
Target Database: C:\extracted\Active Directory\ntds.dit

Defragmentation Status (% complete)

 0  10  20  30  40  50  60  70  80  90 100
|---|---|---|---|---|---|---|---|---|---|
.....

Copying registry files...
Copying C:\extracted\registry\SYSTEM
Copying C:\extracted\registry\SECURITY
Snapshot {2d42528d-9ce6-43a6-a16a-95a77d70f966} unmounted.
IFM media created successfully in C:\extracted
ifm: quit
ntdsutil.exe:

```

Figure 4.2: NTDS.dit, SYSTEM and SECURITY extraction using *NTDSUtil*

file. Once the shadow copy has been successfully created, the user can execute the Windows copy application to copy the NTDS.dit file to a folder of choice. The Windows copy command can be used to extract the *SYSTEM* file from the created volume shadow copy. Another way to extract the *SYSTEM* file is to use Microsoft registry⁷.

Listing 4.1: Example extract from *secretsdump*

```

root@kali:~/Desktop# python /usr/share/doc/python-impacket/examples/secretsdump.py
-ntds ntds.dit -system sys -history -just-dc-ntlm -outputfile results local
Impacket v0.9.17-dev - Copyright 2002-2018 Core Security Technologies

```

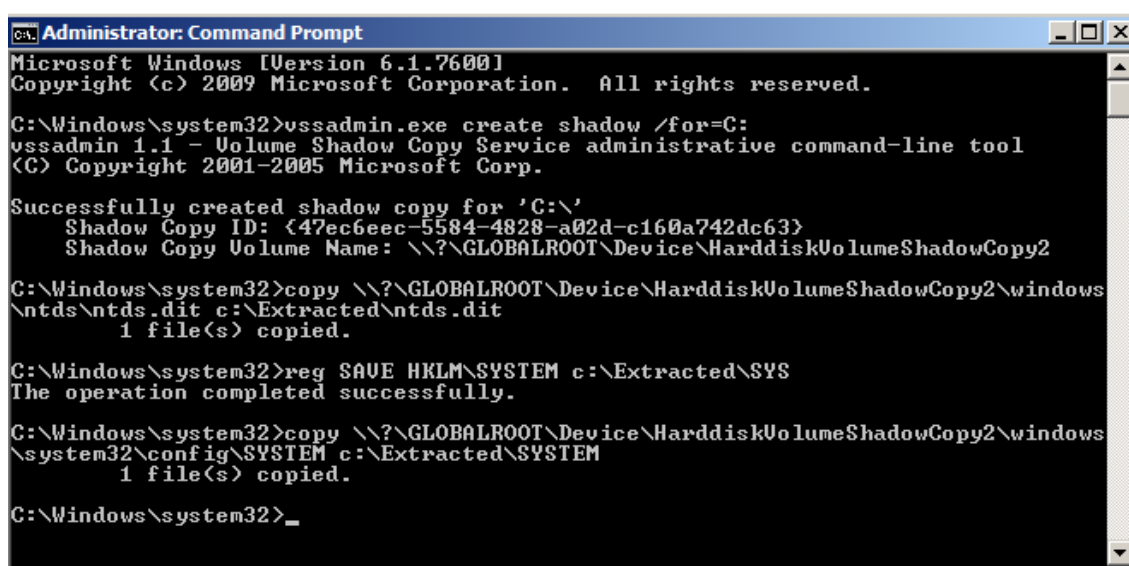
```

[*] Target system bootKey: 0x55a50b845d82c29825f8113f407a6682
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: 4c1ad6c7602e71d02b1472a5b71fa1e2
[*] Reading and decrypting hashes from ntds.dit
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
TestUser:1104:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::

```

Listing 4.1 shows an extract of the NTDS.dit file using the *secretsdump.py* script. For *secretsdump.py* to extract NTLM and LM hashes the user must supply the Python script

⁷[https://msdn.microsoft.com/en-us/library/windows/desktop/ms724871\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms724871(v=vs.85).aspx)



```
Administrator: Command Prompt
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>vssadmin.exe create shadow /for=C:
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2005 Microsoft Corp.

Successfully created shadow copy for 'C:\'
Shadow Copy ID: {47ec6eec-5584-4828-a02d-c160a742dc63}
Shadow Copy Volume Name: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2

C:\Windows\system32>copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2\windows
\ntds\ntds.dit c:\Extracted\ntds.dit
1 file(s) copied.

C:\Windows\system32>reg SAVE HKLM\SYSTEM c:\Extracted\SYS
The operation completed successfully.

C:\Windows\system32>copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2\windows
\system32\config\SYSTEM c:\Extracted\SYSTEM
1 file(s) copied.

C:\Windows\system32>_
```

Figure 4.3: NTDS.dit and SYSTEM extraction using *Vssadmin*

with the following arguments:

- ntds - the location of the NTDS.dit file
- system - the location of the SYSTEM file
- history - extracts the old password hashes of the user if they are contained within the database
- just-dc-ntlm - extract only NTLM
- outputfile - dump hashes into a file called results
- local - indicates to *secretsdump.py* that the NTDS.dit file is local to the system and not a remote connection.

Once the information needed has been received, *secretsdump.py* decrypts and reads the hashes from the NTDS.dit file. It then stores the hashes into a user-defined file for further analysis.

4.2 Extraction of hashes

Prior to cracking the hashes the researcher had to remove all information that was not required for the research. The following information is displayed by Impacket's *secretsdump* python script and needs to be removed:

- username
- SID
- kerberos keys
- computer accounts

The username and SID were removed as this research did not assess any aspect of the username or the quality of passwords for users or groups that have the same SID. Kerberos keys and computer accounts are also removed as kerberos keys do not store any passwords and computer accounts do not fall within the scope of this research.

Once the hashes were separated the script given as Listing 4.2 was run on each of the files:

Listing 4.2: Python hash counter script

```

#! C:\python27
from collections import Counter
import csv

cnt = Counter ()
write_file = "output.csv"

with open(write_file,"w") as output:
    for line in open('combined_ntlm.txt', 'r'):
        for word in line.split():
            cnt[word] += 1
    for k,v in cnt.most_common():
        output.write( "{} , {} \n".format(k,v) )

```

The script has been uploaded to Github⁸ for easier public access. The script was created to count the occurrence of LM and NTLM hashes that were extracted from the NTDS.dit databases and output the count for each hash to a Comma-Separated Values (CSV) file.

Table 4.1: Hashes collected

Hashes collected	
LM	46807
NTLM	170433

⁸https://github.com/ActivateDZA/thesis/blob/master/hash_counter.py

Table 4.1 shows the count of the collected hashes. A total of 46807 LM hashes and 170433 NTLM hashes were extracted from the NTDS.dit files. The two numbers are vastly different because some organisation had disabled the LM hashing algorithm from being used internally because of the security issues of how it hashes the password as described in Chapter 2 and illustrated in Figure 2.10.

An unexpected finding in one of the NTDS.dit database files was that one organisation was storing user passwords using reversible encryption. The passwords that were stored in a reversible encryption were added to cracked password list to be analysed along the passwords that are cracked. According to Microsoft (2008) Microsoft AD allows administrators to enable the storing of passwords using reversible encryption because it allows users to use Microsoft AD for password storage and the legacy applications can use Microsoft AD to authenticate the user.

Microsoft AD by default has this policy disabled and Microsoft does not recommend it be enabled. The reason one organisation had it enabled, could be because of legacy applications that require user passwords to be sent in clear text from Microsoft AD.

While applications may want to use the clear text password for authentication purposes it is not recommended as if someone with malicious intent has access to the Microsoft AD NTDS.dit database file they will be able to extract all the passwords stored there and be able to view them without having to crack them. There have been cases that have come out of organisations that stored customer passwords in clear text such as the Australian Tax Office⁹, T-Mobile¹⁰ and Hetzner South Africa¹¹.

4.3 LM hashes

As shown in Table 4.1 46807 LM hashes were collected. Using the Python script in Listing 4.2 the LM hashes collected were analysed. The hash value *aad3b435b51404eeaad3b435b51404ee* had the highest frequency at 44194 occurrences. This LM hash is blank which is the LM hashing algorithms way of interpreting, no password was created and therefore was excluded from any further analysis. The final count of LM hashes collected after excluding the blank password was 2614. This exclusion process was done so that the

⁹<https://www.itnews.com.au/news/ato-passwords-stored-in-clear-text-334921>

¹⁰https://motherboard.vice.com/en_us/article/7xdeby/t-mobile-stores-part-of-customers-passwords-in-plaintext-says-it-has-amazingly-good-security

¹¹<https://www.htxt.co.za/2017/11/03/hetzner-stored-passwords-in-plain-text-to-make-customer-support-easier/>

results did not over inflate the LM hash count and therefore give incorrect results regarding user LM passwords. What was interesting was that five of the NTDS.dit database files did not use the LM hashing algorithm while in the remaining seven NTDS.dit databases, all the users had LM hashes. Only the first five characters of the LM hashes are shown to protect the password hashes from being exposed by this research.

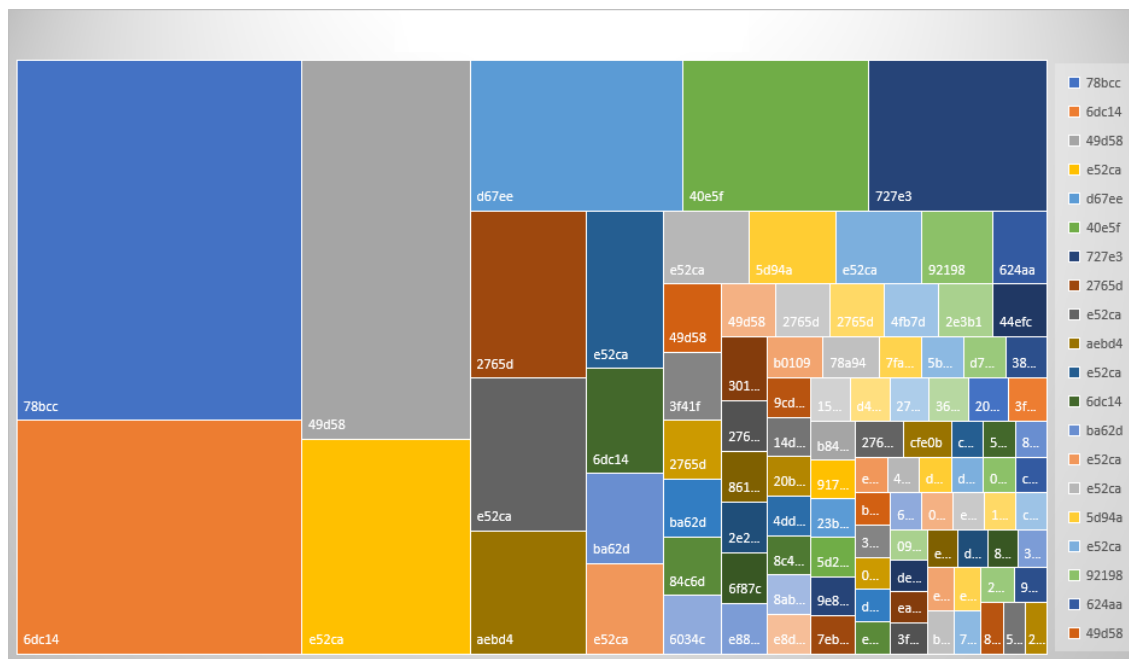


Figure 4.4: LM hash treemap top 100

Figure 4.4 shows the treemap of the top LM 100 hashes while Figure 4.5 shows the top ten LM hashes obtained. In Figure 4.5, the LM hash starting with *78bcc* occurred 179 times and analysing this further, it was found that this hash value appeared 177 times in one NTDS.dit database and one time in two other NTDS.dit databases. The second highest LM hash count 116, was for *6dc14* followed closely by that for *49d58* namely, 111.

Given the very high count of the top three LM hashes it is possible that these are generic passwords that were created by the administrators for the users to use once on logon and then change. But if the administrator does not set the *Change on first logon* flag on the users profile, the user may continue to use that password as their own password. Another reason for the high LM hash count could be that the organisation may have used Microsoft AD when LM hashes were still widely used and accepted. But when the major vulnerability with LM appeared, the organisation did not disable the LM hash from being used because the administrator may not have known about the risks of using LM or the administrators may not given it much attention because AD is an internal system and not externally facing.

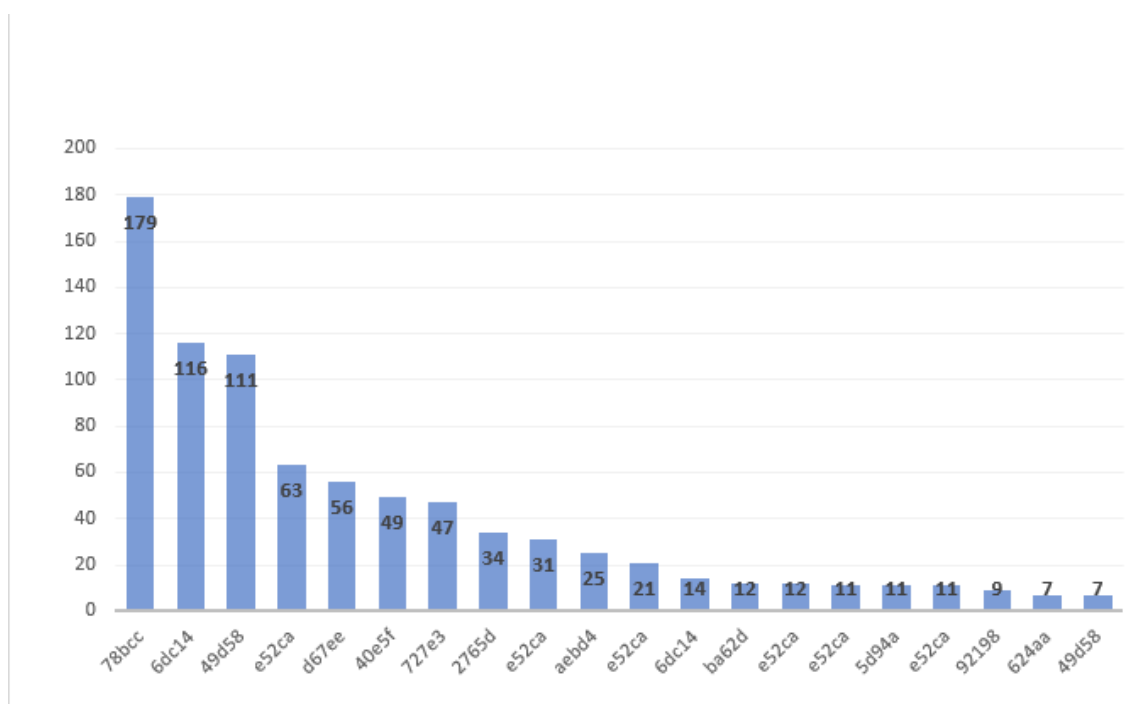


Figure 4.5: LM hash top ten

Table 4.2: Count of least frequently appearing LM hashes

Unique hash appearance	Frequency n = 79
2 times	47
3 times	25
4 times	7

Another interesting observation of the top LM hash and the third most frequent LM hash in Figure 4.5 was that they were both found in one NTDS.dit database while the second most frequent LM hash was found in another single NTDS.dit database. The remaining LM hashes in the top ten list were spread across the seven NTDS.dit databases. LM hashes that were unique and only appeared once totalled 1523 of the hashes which translated to a overall percentage of 58%. This means that more than half of all the LM hashes collected, excluding the blank LM hash represented unique passwords.

Table 4.2 shows the count of least frequently appearing LM hashes; 47 LM hashes appeared only twice in all the NTDS.dit databases, 25 appeared three times and seven hashes appeared four times.

4.4 NTLM hashes

A total of 170433 NTLM hashes were collected. This NTLM hash count excludes computer accounts as according to Microsoft¹² domain-joined computers will have an NTLM hash. As stated above the reason for excluding computer password NTLM hashes is because we were not interested in assessing the quality of passwords generated for computers; only human-created passwords were assessed. Figure 4.6 shows the top 20 NTLM hashes found

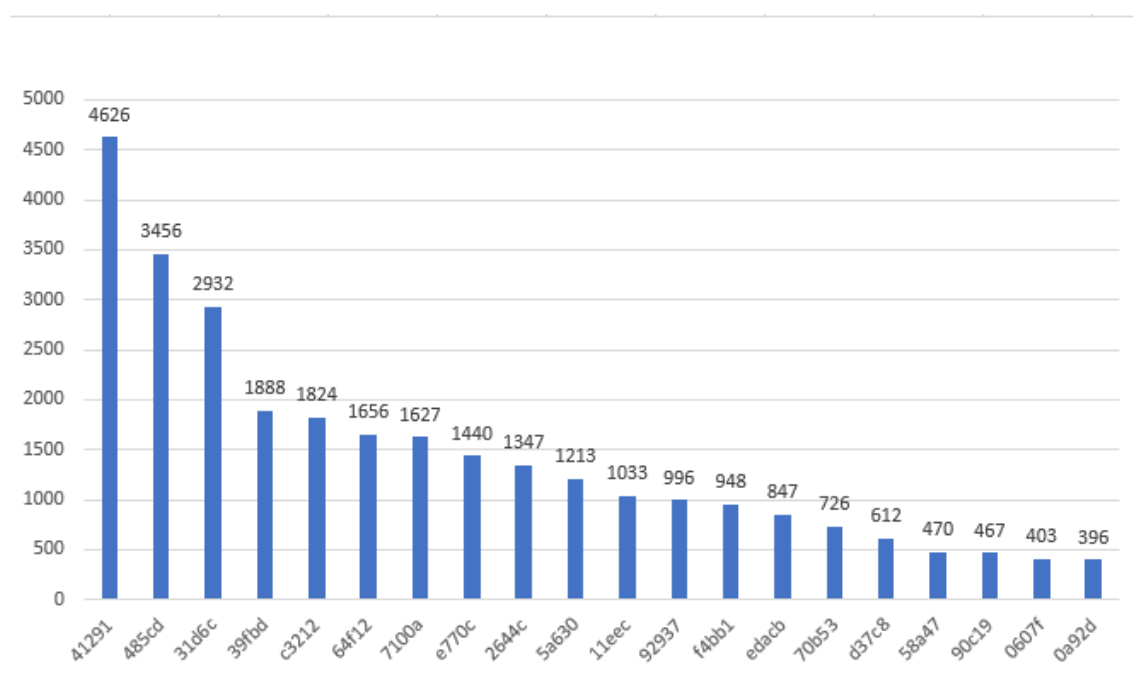


Figure 4.6: NTLM hash top 20

in the NTDS.dit database files. The NTLM hash starting with *41291* was the top NTLM hash found with a count of 4626, followed in second place by *485cd* with 3456 occurrences and *39fbd* in third place with 2932. In the NTDS.dit databases that were assessed the top NTLM hash accounted for 2.7% of the total NTLM hashes collected. There were 76355 unique NTLM hashes found in the NTDS.dit database files, which accounted for a total of 45% of all NTLM hashes assessed. While this number is below 50% it is encouraging that 45% of all the NTLM password hashes of user accounts were unique and not shared even across the different NTDS.dit databases. Other NTLM hashes that had low counts between two and four, are shown in Table 4.3. A total of 13576 hashes only appeared twice in all the NTDS.dit database files, 1513 NTLM hashes appeared three times and 652 NTLM hashes appeared four times.

¹²<https://blogs.technet.microsoft.com/askds/2009/02/15/machine-account-password-process-2/>

Table 4.3: Count of least frequently appearing NTLM hashes

Unique hash appearance	Frequency n = 15741
2 times	13576
3 times	1513
4 times	652

4.5 LM hash attack

Hashcat can attack LM hashes. To attack LM hashes the hashcat application was supplied with a few parameters. As discussed in Section 2.4.2, the LM hash has a major weakness in the hashing algorithm where it capitalises all the characters and then breaks the password into two 7-character long sections.

The following command was run against the LM hashes collected:

```
E:\User>hashcat64.exe -a 3 -m 3000 "E:\hashes\LM\Combined_LM.txt" -o  
"E:\hashes\hashcat-4.1.0\cracked\Combined_LM_cracked.txt"
```

Where:

- *hashcat64.exe* is the hashcat binary,
- *-a 3* represents attack mode 3, which is the brute-force attack
- *-m* represents the mode that hashcat must use
- *3000* — is the LM hash mode number in hashcat
- *E:\hashes\LM\Combined_LM.txt* — the location of the LM hashes extracted from the NTDS.dit database files
- *-o* — is the output file where cracked hashes are stored.

As the LM hashing algorithm has a vulnerability, we did not supply a wordlist to assist in attacking the LM hashes, but instead the brute force method was used to attack the hashes. As stated, there were 2614 LM hashes that were extracted and using the brute force attack we were able to obtain an additional 2242 LM hashes. The whole attack took 25 minutes and 49 seconds to complete.

Because of the weaknesses in the LM hashing algorithm, we were able to obtain 86% of all the LM hashed passwords.

4.6 NTLM hash attack

Table 4.4: NTLM passwords found using wordlists and rules

Wordlist	Rule	Unique passwords found
have_I_been_pwned	–	10409
Breachcompilation	–	1202
weakpass_2a	–	4257
have_I_been_pwned	InsidePro-HashManger.rule	35034
have_I_been_pwned	dive.rule	8060
have_I_been_pwned	_NSAKEY.v2.dive.rule	2804
Breachcompilation	InsidePro-HashManger.rule	706
weakpass_2a	_NSAKEY.v2.dive.rule	14116

Attacking NTLM hashes required more work and attention than the LM hashes. While LM hashes can efficiently be attacked using a brute force method, this is not as efficient for NTLM hashes and not the best approach. Only 40% of the NTLM hashes were found using brute force.

Given the low success rate using straight brute force, a mask attack was used to remove the easy-to-guess passwords. The following command was run on the NTLM hashes:

```
E:\User>hashcat64.exe -a 3 -m 1000 "E:\hashes\LM\_ntlm.txt" -o "E:\hashcat-4.1.0\cracked
\thesis_ntlm.txt" - - increment - 1 ?u?d?s ?1?1?1?1?1?1?1
```

The brute force command is similar to that used for the LM hashes except for the following:

- *m 1000* — NTLM hashes are hash mode 1000
- *increment* — start from 1 and increment to 7 characters long
- *1 ?u?d?s* — apply uppercase, digit and special characters
- *?1?1?1?1?1?1?1* — maximum password length

This mask attack starts from one character, upper-cases the character and adds in a digit and a special character. After the mask attack was completed, a rule based attack was conducted on the NTLM hashes using various wordlists.

The wordlists that were used in conjunction with the NTLM hashes and rules were downloaded from various websites such as *weakpass.com* and *github.com*. The first wordlist

that was used was the *breachcomplication*. Table 4.4 shows the breakdown of each of the various wordlists and rules that were used. To start the wordlist attack, the researcher randomly picked the *have_I_been_pwned* wordlist without the rules to see how many NTLM passwords it could crack. It was able to obtain 10409 passwords in 50 seconds. This was followed by the *breachcompilation* wordlist that had more words than the *have_I_been_pwned* wordlist, and a further 1202 NTLM passwords were obtained in 1 minute 54 seconds. Lastly the researcher ran the final wordlist *weakpass_2a* which was the biggest wordlist of the three. This wordlist was able to obtain a further 4257 passwords.

Applying the rules to the three wordlists gave more NTLM passwords. Starting with the *have_I_been_pwned* wordlist and applying the *InsidePro-HashManager* rule allowed the researcher to obtain a further 35034 NTLM passwords. This was followed by the *weakpass_2a* wordlist when applied with the *_NSAKEY.v2.dive* rule, and it was able to obtain a further 14166 NTLM passwords. Because of the size of the wordlist and the number of rules this ran for 2 weeks resulting in 14166 additional passwords.

4.7 Characters for LM and NTLM hashes

Listing 4.3: Python character counter script

```
from collections import Counter

with open('CombinedPasswords.txt') as f:
    c = Counter()
    for x in f:
        c += Counter(x.strip())
print c
```

Analysing the characters of the clear-text passwords is important to understand the frequency of each character and whether users prefer to use lowercase or uppercase characters. To get a count of the characters of the clear-text passwords the Python script given in Listing 4.3 was used to count all the individual characters and numbers.

The Python script starts by opening *combinedPasswords.txt* which contains LM and NTLM clear text passwords and counting each letter, number and special character that is found. The python script then displays the results to the analyst. The Python script can be downloaded from the researcher's repository on Github.com¹³.

¹³https://raw.githubusercontent.com/ActivateDZA/thesis/master/character_count.py

4.8 Summary

In this chapter we discussed the LM and NTLM hashes obtained and how they were distributed across the NTDS.dit databases. This chapter also discussed the extraction process of the hashes and what information was removed for this research as it did not have any impact on the end results. A Python script was used to give a count of how many LM and NTLM hashes were found and extracted to a CSV file. Once the blank LM password hashes were excluded we found that there were 2614 LM hashes and 170433 NTLM hashes collected. This shows that the majority of password hashes use the NTLM hashing algorithm, which does not have the weaknesses of the LM algorithm.

This chapter also discussed the password cracking tool used called Hashcat. Using Hashcat 86% of the LM password hashes were cracked but only 45% of the NTLM hashes. This chapter also discussed the attack methods used on the LM hashes and the wordlists and rules used on the NTLM.

Chapter 5

Analysis and discussion of organisational passwords

This chapter analyses the LM and NTLM passwords that were cracked in Chapter 4 to determine user behaviour when selecting passwords. Analysis includes the passwords, base words, password lengths and other aspects thereof.

5.1 Cracked passwords

The LM hashing algorithm has a serious hashing vulnerability (see Figure 2.10) where the password is uppercased and split into 7-byte chunks before being encrypted using DES, concatenated and stored. This vulnerability allowed the researcher to crack 2612 LM hashes out of a total of 2614 LM hashes. This accounted for a 99.92% successful crack rate against LM based passwords from the 12 NTDS.dit databases. It was found that out of the 2612 passwords found, 1634 (62.5%) were unique passwords. This is a high number of unique passwords which is promising but the fact that the LM hashing has a major weakness causes all of those passwords to be weak as well. As stated a total of 170433 NTLM hashes were obtained and only 76588 were successfully cracked. This accounts for 45% of the NTLM hashes.

5.2 Top ten passwords

The top ten passwords that were found are listed in Table 5.1. All the passwords found in Table 5.1 are weak passwords but five of the passwords are even weaker because they appear in the 25 weak password list that started in 2011. According to David (2011) and Olivia (2012), *ABC123*, *ABC12345*, *PASSWORD*, *PA\$\$WORD*, *PASSWORD01* and by extension *pass@word1* are weak passwords that have been documented as such since 2011 but still appear as passwords in corporate South Africa in 2017 and 2018.

Table 5.1: Top ten passwords found

Password	Count (n=70751)	Frequency (%)
ABC123	179	0.25
FISHEAGLE1\$	116	0.16
ABC12345	111	0.16
Password01	91	0.13
PASSWORD	63	0.09
<REDACTED NAME>123 (Company E)	56	0.08
pass@word1	51	0.07
PP<REDACTED NAME>123 (Company D)	49	0.07
PA\$\$WORD	47	0.06
<REDACTED NAME>00 (Company B)	34	0.05

Analysis of the top ten passwords shows that three of these contained company names and in fact some users from each of the organisations analysed created passwords that contained the company name. In addition, users also created passwords that contained product names, locations (cities and town) and department dependent passwords like *SQL_SQL_30*. This is a big risk for the organisation because these passwords are words that malicious actors might start with when trying to guess the user password. The malicious actor can use freely available tools such as *CEWL*¹ which can spider a target website and generate a custom word list based on the unique words that are found on the website.

5.3 Top ten base words

A base word is a word that cannot be broken down any smaller, for example "*careless*" can be broken into two words, *care* and *less*. According to Merrell (n.d.), a base word can

¹<https://digi.ninja/projects/cewl.php>

have a prefix and a suffix which are letters added at the beginning and end of the word, respectively. Table 5.2 illustrates the top ten base words found in the combined cracked passwords. *"Password"* which is the top weakest password as stated by David (2011) and

Table 5.2: Top ten base words

Base word	Count (n=70751)	Frequency (%)
PASSWORD	1509	2.13
<REDACTED NAME>(Company A)	1354	1.91
<REDACTED NAME>(Company B)	975	1.38
<REDACTED NAME>(Company C)	574	0.81
<REDACTED NAME>(Company D)	369	0.52
pa\$\$w0rd	299	0.42
july	296	0.42
april	292	0.41
summer	259	0.37
june	254	0.36

Olivia (2012) is also the top weakest base word in the combined cracked password list. The next four top base words are company names, which indicates that users in these companies frequently use company names as passwords. Combining the top ten used company names, they account for 4.62% of all the password base words. Months and seasons are also popular in the cracked password list. As discussed in Chapter 2, some organisations force users to change their passwords on a monthly basis and therefore users become creative in the password creation process by using months and seasons as their base words.

The users were creative in the way they were using months as part of their passwords. For example users used the short hand version of months like January *"Jan"*, February *"Feb"* and March *"Mar"*. Analysing the cracked password list we found that *"Jan2017"* appeared 85 times, *"Feb2017"* appeared 35 times, *"Mar2017"* appeared 39 times and *"apr2017"* appeared 33 times. There was a lesser chance that a user would create the same password multiple times within the same month so one can reason that some users shared similar password creation behaviour.

The one factor that differentiated the users, was that users appended different special characters at the end of the password. According to Van Allen (2013), people choose passwords according to readily available information around them so words like *password*, *july* and *summer* might be used by people to generate passwords.

Other possible base words that did not appear in the top ten base word list according to Van Allen (2013) are:

- names and surnames
- countries, cities and towns
- days
- animals
- favourite sports team
- company made products
- and Bible verses.

Of these, names and favourite sports teams appeared in the cracked password lists. Analysing the names that appeared in the cracked password list we found the following: Diane (English), Joaan (Afrikaans), Thato, Thando (Zulu), Kananelo (Sotho) and Aphiwe (Xhosa) amongst others. South African surnames were also among the words that appeared in the cracked password list, for example Smith (English), Zulu (Zulu), and Mabitha (Sotho).

According to Schoeman (2017) the 2001 South African census revealed that 79.8% of all South Africans that are religious identified themselves as being Christian and as a result, we found several Bible verses in the cracked password list, including verses from Deuteronomy (one occurrence), Ephesians and Proverbs (3 occurrences each). The highest number of occurrences were verses from Joshua (51 times), Hebrews and Revelations (7 and 9 occurrences respectively).

One user used the password "*John14:1*" and only changed the last number of the password. This user used the same password creation process 11 times and the last password that was obtained was "*John14:11*". The list below shows the passwords that users used that were direct bible verses:

- John 14:1 => appearing as *John14:1*
- Hebrews 1:35 => appearing as *Hebrews1:35*
- Luke 13:7 => appearing as *Luke13:7*

- Revelations 3:20 => appearing as *Revelations3:20*
- Matthew 22:39 => appearing as *Matthew22:39*
- Romans 8:8 => appearing as *Romans8:8*
- Psalm 150:6 => appearing as *Psalm150:6*
- First Peter 5:5 => appearing as *1Peter5:5*
- Second Corinthians 3:15 => appearing as *2Cor3:15*
- First Timothy 1:15 => appearing as *1Timothy1:15*
- Deuteronomy 7:18 => appearing as *1Deut7:18*

The *John14:1* up to *John14:11* passwords seem to be created by one person because the only thing that changes in the password is the last two digits. The other listed Bible passwords all follow a similar password creation pattern as *John14:1*.

5.4 Keyboard walking patterns

In Chapter 2, keyboard walking was discussed where some users may use it to create complex passwords. Li *et al.* (2014) illustrated three keyboard pattern techniques that Chinese and English users employed. The main issue with keyboard walking is that computer systems can easily recreate the same patterns that the user is creating as the user defined pattern password must be easily remembered by the user.

Analysing the cracked password list using a simple west-to-east keyboard walking for text generated some interesting results; for example, searching for passwords that contained the word *qwerty* returned 115 matches. The word *qwerty* is a simple west-to-east directional password. Other west-to-east transversals were *asdfg* which gave 40 matches and *zxcvb*, with 38 occurrences. Checking for passwords that moved in an east-to-west direction such as *ytrewq* only gave eight matches compared to the west-to-east direction with 115 matches, *gfdsa* gave a lower count of ten matches and *bvcxz* resulted in 11 matches. Checking for numeric west-to-east keyboard walking, *12345* gave a count of 2720 passwords that contained the numbers and checking for *1234567890* gave 31 matches of passwords that contained these numbers. Checking for east-to-west direction for numeric *0987654321* returned only 4 matches.



Figure 5.1: Cracking passwords using keyboard walking

Figure 5.1 illustrates one example of the keyboard walking that was tested. The password was created using an easy to remember pattern of using the directions that were discussed in Section 2.3.5. The password was generated using this directional formula

$START \rightarrow (base-character = 1) + 3 * SOUTH + 1 * EAST \rightarrow 3 * NORTH$ to create *1qazxsw2*. There were five matches for the *1qazxsw2* password.

Another example of keyboard walking using the centre of the keyboard was *5tgbnhY^*. The user followed the same direction as Figure 5.1, but instead of adding a number at the end, they added a special character. Six passwords contained similar patterns to *5tgbnhY^*. The users either replaced the character five with % or replaced the special character ^ with the number 6.

Another keyboard pattern technique that Li *et al.* (2014) discussed is using three characters in parallel, for example *zxcasdqwe*. This pattern was used by one user in the crack password list, while six users used *qweasdzxc* as their keyboard pattern. There are users who used keyboard walking on the keypad as illustrated in Figure 5.2. This figure does not illustrate all the various keypad walking techniques of users but illustrates some of the creative walks that were used to create passwords. There were nine users who did keypad walking by including *123698741* into their passwords. This pattern is illustrated in Figure 5.2 as the light blue colour. Other patterns that were of interest were users who employed diagonals for the numbers in their passwords. The grey and orange patterns in Figure 5.2 shows the two diagonal patterns found. The grey pattern was unique and there was only one user who employed this pattern to create a password while the orange pattern had three matches for users who used it to create their passwords.

Four users were identified to be using a "snake" like pattern on their passwords. The

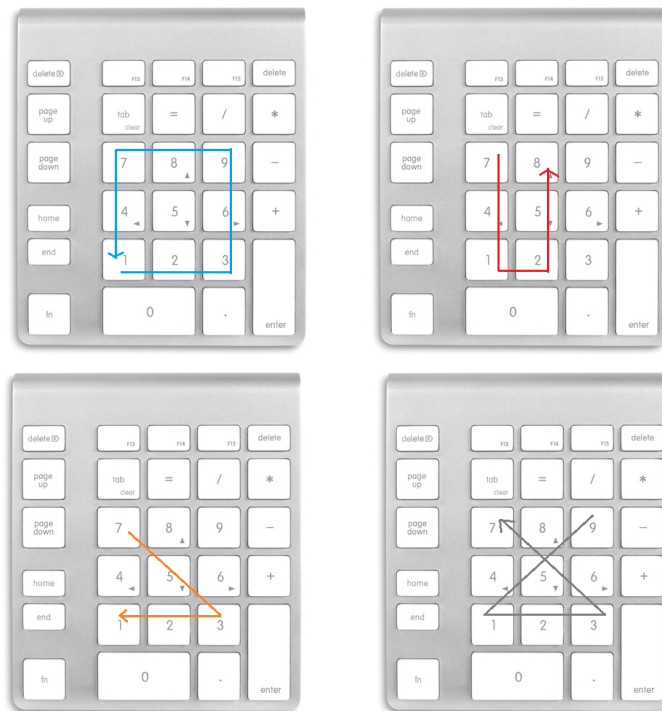


Figure 5.2: Keypad walking patterns used

sequence of numbers, *987456321*, was contained in their passwords along with other text. Looking at the keypad in Figure 5.2, one can see the pattern that was employed by these users. The passwords that contained this pattern ranged in length from 12 characters to 14 characters but all of these are weak because the users employed keyboard walking on the keyboard and on the keypad as well. Fourteen users employed the same "snake" like pattern of *0123456789*.

Some users used their South African ID numbers as their passwords. In the cracked password list there were seven matches for passwords that followed the regex in Listing 5.1.

```
((\d{2}((0[13578]|1[02]) (0[1-9]||[12]\d|3[01]) |(0[13456789]|1[012])
(0[1-9]||[12]\d|30)|02(0[1-9]|1\d|2[0-8])))|([02468][048]||[13579][26])
0229))(( |−)(\d{4})( |−)(\d{3})|(\d{7}))
```

Listing 5.1: Regex to identify South African ID numbers obtained from: http://www.regexlib.com/REDetails.aspx?regexp_id=1573

The regex in Listing 5.1 is able to find all South African ID numbers that match the following ID format variations:

1. 200101 4800 086,
2. 200101-4800-086,
3. and 2001014800086

The main issue with using the South African ID number as a user password is that it is easy to guess using a similar regex to that given above. A malicious user can generate many random ID numbers to crack such a password. Additionally exposing the ID number may also expose much personal information.

5.5 Password length analysis

Password lengths are very important in the creation of passwords since having a longer password can make it harder for a malicious user to crack that password if it is stored in a secure manner. Password lengths have been discussed in Section 2.2 where CIS guidelines state that password lengths be set at a minimum of 14 characters while Grassi *et al.* (2017) and the NIST 800-63-3 guideline state that password lengths should be set at a minimum of eight characters.

Table 5.3 shows the top ten password lengths in the cracked password list. Passwords that are eight characters in length have the highest count of 20463 accounting for 28.92% of all passwords. This count can be attributed to the recommendations that are given by Microsoft (n.d.a) who state that minimum password lengths should be set to 8 characters and at 14 characters for greater security. The second highest password length was nine characters with 17124 occurrences, accounting for 24.2% of all passwords.

After the second highest password length the next four password lengths all followed sequentially starting at 10 characters up to 13. The password length that is recommended by CIS and Microsoft (n.d.a) only ranked ninth in the password length top ten. According

Table 5.3: Top ten password lengths

Password Length	Count (n=70750)	Frequency (%)
8	20463	28.92
9	17124	24.2
10	13373	18.9
11	8119	11.48
12	4499	6.36
13	2159	3.05
7	1706	2.41
6	1197	1.69
14	1086	1.53
15	320	0.45

to Johansson (n.d.) the maximum NTLM password length is 127 characters and in the cracked password list there were passwords that are over 100 characters long. These passwords were found in the password database that stored passwords using reversible encryption. One password was 119 characters long and surprisingly 26 passwords were 120 characters and 59 passwords were 127 characters in length. These passwords were not cracked but because they were stored in Microsoft AD using reversible encryption, the passwords are considered weak if someone has access to the password database. The passwords are long and contain no easily recognisable pattern and thus it is assumed they were not manually created by a user. One can also assume that the passwords may belong to the user but were generated by other means such as a password generator.

A total of 1486 passwords or 2.1% of all passwords ranged from one character to six characters while 23655 or 33.43% of the passwords were eight characters in length. Since 47095 or 66.57% of all passwords analysed were 8 characters or more, the majority of all passwords was over eight characters in length. This finding corroborates the finding in Section 2.2.1 where organisations in their password policies state that users must create passwords that are at least eight characters in length.

5.6 Analysis of digits used

Password policies discussed in Section 2.2 indicate that passwords must include a letter, number and in some cases a special character. Users normally add numbers to their passwords to make them more complex but the problem with this is that users either follow patterns or include dates. Table 5.4 illustrates the top ten sequences of numbers for various digit positions. This shows that users employ keyboard walking or use adjacent numbers to append to passwords in order to remember their patterns. In the cracked password list, the passwords that ended with numbers were assessed to identify how users used patterns in the numbers.

Table 5.4 shows that all of the top ten last five digits were patterns. It comes as no surprise that *12345* was the top number sequence considering that Symantec (2010) had *123456* as its top weak password. Analysis of last 4 digits as illustrated in Table 5.4(b)

Table 5.4: Sequences of digits appended to passwords

5.4(a) Last 5 digits

Number	Count	Frequency (%)
12345	1387	1.96
23456	579	0.82
56789	190	0.27
34567	172	0.24
54321	155	0.22
45678	150	0.21
00000	55	0.08
11111	44	0.06
98765	40	0.06
01234	38	0.05

5.4(c) Last 3 digits

Number	Count	Frequency (%)
123	3487	4.93
017	1588	2.24
345	1497	2.12
234	1271	1.8
016	1200	1.7
456	967	1.37
789	561	0.79
015	488	0.69
321	484	0.68
000	380	0.54

5.4(b) Last 4 digits

Number	Count	Frequency (%)
2017	1550	2.19
2345	1447	2.05
1234	1182	1.67
2016	1158	1.64
3456	601	0.85
2015	456	0.64
6789	243	0.34
4321	219	0.31
4567	216	0.31
2014	206	0.29

5.4(d) Last 2 digits

Number	Count	Frequency (%)
23	4002	5.66
17	2339	3.31
16	1868	2.64
01	1778	2.51
45	1765	2.49
12	1612	2.28
34	1494	2.11
11	1195	1.69
00	1149	1.62
56	1145	1.62

shows that years are popular with users when adding in numbers with 2017, 2015 and 2014 all featured in the top 10. This research did not go into as much detail in trying to identify date format like that discussed in Section 2.5 by Li *et al.* (2014) but we can see that South African users also employ similar password generation techniques. These numbers are popular with users when creating passwords because they are easily recalled by the user. The other numbers in Table 5.4(b) are keyboard walking examples since users tend to create easily memorisable numbers to append to their passwords. Users also tend to use and append the number zero in their password as shown in the Table 5.4, 5.4(c), 5.4(d).

5.7 Use of characters in passwords

Understanding the frequency of characters that are used to create passwords provides information on which characters are important to users. Using the Python script in Section 4.7, all the characters in the cracked password list were counted. The Python script counted 96 different characters, which is the default number of characters on a US qwerty keyboard. Figure 5.3 illustrates the top 20 characters found and their occurrence in the cracked password list.

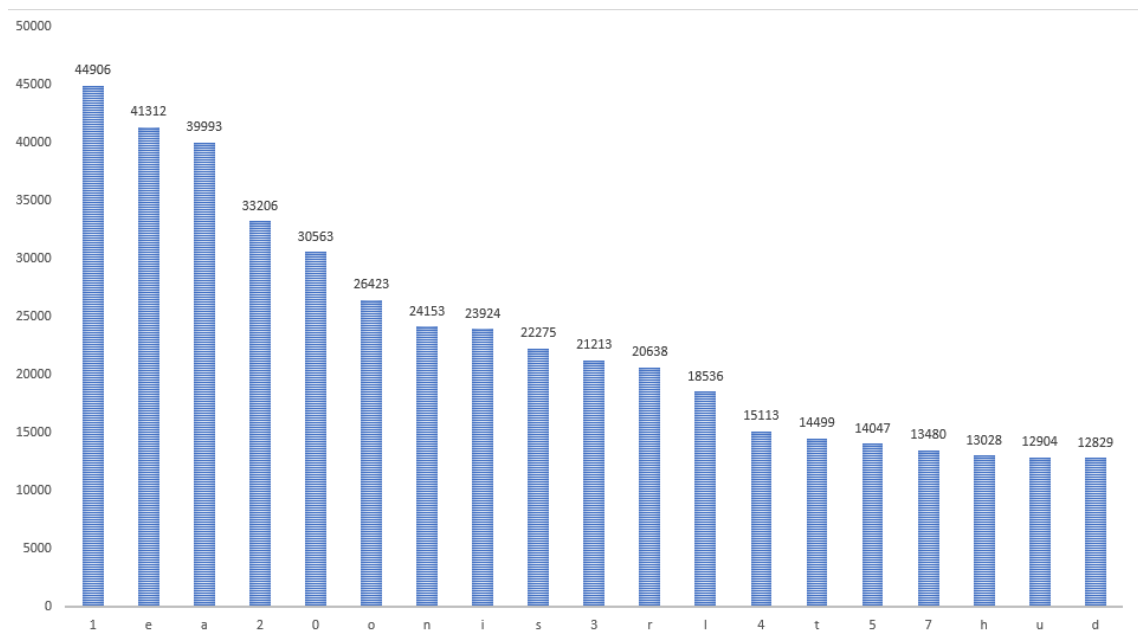


Figure 5.3: Top 20 character counts in cracked password list

The character *1* appears the most in all the analysed passwords followed by the character



Figure 5.4: Top 20 character heatmap where darker colours show higher frequency of use

e and in third place character *a*. Analysing the top eight characters further, the top four characters are located on the left hand side of the keyboard while the next group of four characters is located on the right hand side of the keyboard. Generating a heatmap of the top 20 characters illustrates how popular the character *l* is and this is supported by looking at how often the number *1* appeared in Table 5.4. This is further supported by reviewing the keyboard walking section and how users apply keyboard walking when appending numbers to their passwords for example *12345*. Figure 5.4 illustrates how the top 20 characters lean to either the left or right hand sides of the keyboard while the centre characters, for example *fgbvjym* are not predominant. Figure 5.4 also shows how the top number characters follow the keyboard walking pattern. The character number *1* is the top character in the list followed by the number *2* and *0* in third place. Other numbers that featured in the top 20 are *3, 4, 7* and the only one that is missing in the top 20 is the number *6* which is 21st out of the possible 96 characters. The English language vowels consist of *a, e, i, o, u* and sometimes *y*². According to Figure 5.3, vowels *e, a, o* and *i* feature predominately in the top ten with characters *s* and *n* being the only non-vowel letters in the top ten.

5.8 Letter frequency analysis

Cornell (2004) analysed 40,000 words and found that the letter *E* has the highest letter frequency with a frequency occurrence of 12.02%. It was followed by the letter *T* with a

²<https://www.merriam-webster.com/dictionary/vowel>

frequency occurrence of 9.10% and the letter *A* with 8.12%. Figure 5.5 shows the English letters that Cornell (2004) analysed and Figure 5.6 shows the cracked password letter frequency. The English letters are represented with the blue colour and the password letters are represented in the orange colour. The character are sorted according to the English letter occurrence frequency to see if the cracked passwords also follow the same occurrence frequency.

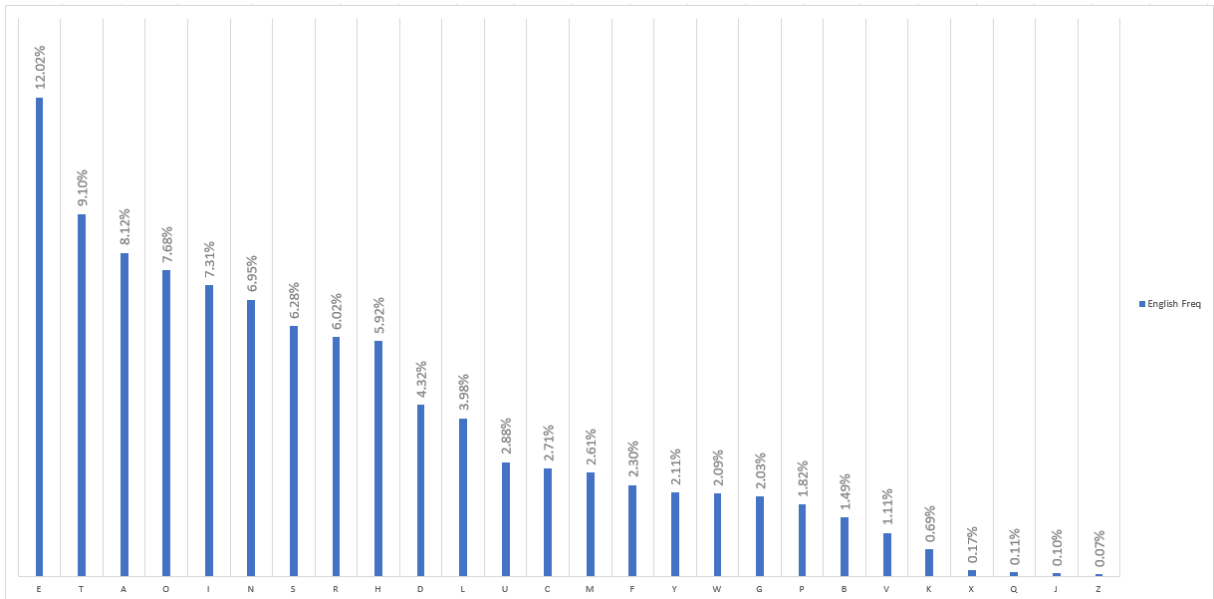


Figure 5.5: English letter frequency

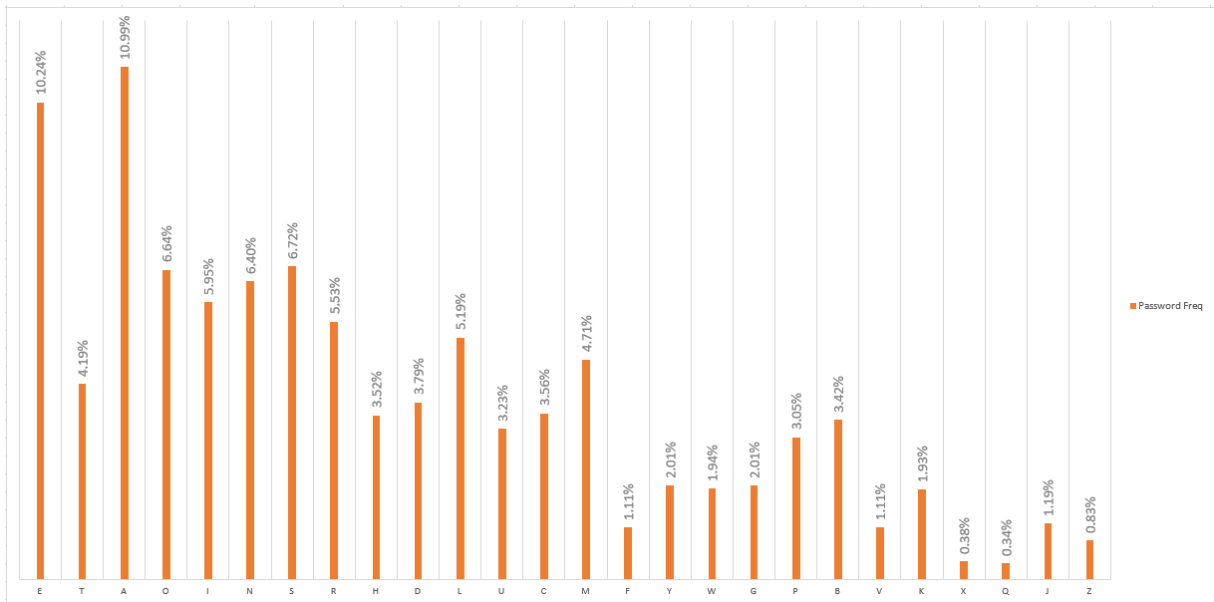


Figure 5.6: Cracked password letter frequency

Analysing Figure 5.5 we notice that the letter *E* has the highest frequency for both English

and the cracked password list usage. Differences start when analysing the second highest English letter frequency, the letter *T*, where in the cracked password list it appears ninth. This is a big difference because the letter *T* is situated in the middle of the keyboard, users may be biased towards letters that are on the far left or far right of the keyboard. The third highest English frequency is the letter *A* but on the cracked password list it appears second in the list.

Further analysis of Figure 5.5 shows that most of the letter frequency percentages for English and cracked password usage are close to each other. There are some exceptions with the following letters having a higher frequency in the cracked password list than their English usage: *A,S,L,U,C,M,P,B,K,X,Q,J,Z*. The letter *A* has an extremely high cracked password frequency which means that users often added the letter *A* in their passwords. There are a few letters that had a similar English and cracked password frequencies. These letters are *Y,W,G* and letter *V* is the only letter to have the same frequency for both its English and cracked password usage at 1.11%.

The vowel letters of *E,A,O,I* dominate the top five as illustrated in Figure 5.5 which is understandable considering how important vowels are in words. For both English frequency and the cracked password list the vowel letters are shown to be of high importance for the English words analysed and the cracked passwords analysed.

5.9 Character set usage

Character sets are important to users that are looking to crack passwords. Understanding the character sets that are being used can allow the user to understand how the majority of the passwords are structured thereby allowing them to attack the passwords more easily. Hashcat uses character sets and masks to assist users to crack passwords as discussed in Section 3.7. Table 5.5 shows all the character sets for all the passwords that were analysed and 47.34% or 33496 out of 70750 of all the passwords that appear in the cracked password list have a character set of *mixedalphanum*. *Mixedalphanum* is a character set that explains passwords that contain letters and numbers. The second place character set was *mixedalphaspecialnum* which came in at 28.64% and accounted for 20262 out of 70750 passwords. The *mixedalphaspecialnum* character set contains letters (upper and lowercase), numbers and special characters in their passwords. In third place *loweralphaspecialnum* accounted for 7.62% or 5392 of all the passwords analysed.

Table 5.5: Character sets for cracked password list

Character set	Count (n=70750)	Frequency (%)
mixedalphanum	33496	47.34
mixedalphaspecialnum	24378	34
loweralphaspecialnum	6368	9
loweralphanum	1974	2
upperalphaspecialnum	1431	2
upperalphaspecial	1235	1
mixedalphaspecial	609	0.86
loweralpha	603	0.85
upperalpha	281	0.4
numeric	138	0.2
upperalphaspecial	123	0.17
mixedalpha	83	0.12
loweralphaspecial	24	0.03
special	4	0
specialnum	3	0

5.10 Summary

In this chapter we discussed the cracked passwords analysing over 70750 clear-text passwords. This chapter also analysed the top ten passwords, and found that the variations of the word *password* were the most common password. It also analysed the base words and found that users when picking passwords often use company names as their base words as it may be easier for them to remember the password. The word *password* is a popular base word while months and Bible verses were also very popular words to use in passwords. Keyboard walking is a useful technique if users are forced to change their passwords every month. They tend to adopt methods that allow them to create new passwords that will still allow them to easily remember their new password.

Chapter 6

Generating and analysing passwords using African languages

In Chapter 5 we analysed the organisational passwords that were cracked. We found that many of the top ten passwords used a variation of the word *password*. The highest frequency base words such as *July*, *Password* and *Summer* come from the English language using this information this chapter considers the creation of passwords using the eight African languages. This chapter analyses the results of the African derived passwords that were uploaded to various password cracking services. The chapter concludes by assessing whether top English passwords that are converted to African languages are more secure than their English counterparts.

6.1 Creation of African language passwords

One of the objectives of this research is to investigate whether converting top weak English passwords to various Southern African languages decreases the possibility of these passwords being attacked in the same way as the English-based passwords. Thus, we set out to determine whether African language derived passwords are more secure than their English counterparts.

Ñ Ð Ľ Ñ Ŧ ñ ð ĩ ñ ĩ

Figure 6.1: Venda special characters¹

¹https://africanlanguages.com/venda/venda_characters.png

Table 6.1: Top weak passwords per language

English	Zulu	Swati	Sepedi	Tshivenda	IsiNdebele	Setswana	Xitsonga	Xhosa
Password	iphasiwedi	Lekungena	Phasewete	Phasiwede	Ipasiwedi	Phasewete	Phasiwede	iphasiwedi
let me in	ngingcise					Mpolele		ndifake / phakathi
welcome	wamukelekile	Mukelekile	Amogetswe	Davhidza	Siyalemukela	O amogetswe	Amukela	Wamukelekile
freedom	inkululeko		bolokolohi					inkululeko
hello	Sawubona	Sawubona	Thobela	Matsheloni Avhudi	Lotjhani	Dumela	Avuxeni	Molweni
football	ibhola	ibhola	bolo	bola		bolo	Bolo ya minenge	ibhola
master	inkosi	Inkhosi	Kgosi	Thovhela	Ikosi	Kgosi	Hosi	Ukumkani
bible	iBhayibheli	iBhayibheli	Bibele	Bivhili		bebele	Bibele	iBhayibhile
fish	ihlanzi		hlapi			tlhapi	Nhlampfi	intlanzi
july	uJulayi	Julayi				Phukwi	Mawuwani	EyeKhalala
summer	ihlobo	Ihlobo	lehlabula			selemo	Ximumu	ehlobo
Come in	Ngena	ngena				kena		Ngena
shadow	isithunzi	sifunti				seriti		isithunzi
monkey	inkawu	inkawu				tsweni		isilwanyana
hunter	umzingeli	liphisi						ngumzingeli
trust no one	ungathembemunutu					setshepi/ motho		suthemba namnye
access	ukungena							fi kelelo
hammer	isando	candvula						nyundo
secret	imfihlo					sephiri		mfihelelo

The Tshivenda language uses special characters such as those illustrated in Figure 6.1; however, special characters were removed from words for this review because we are assessing generated passwords using the QWERTY keyboard and special characters that are normally used within corporate South Africa.

To achieve the objective, weak English passwords were converted to eight African languages, namely Zulu, Swati, Sepedi, Tshivenda, IsiNdebele, Setswana, Xitsonga and Xhosa. The English passwords were selected from the top 500 weak passwords as published by Symantec Connect². Table 6.1 shows the English words (highlighted in yellow) that were selected out of the 500 possible weak password. These words were selected for their ease of conversion to the various African languages. The English words were converted to the various African languages as illustrated in Table 6.1.

In total 198 words were converted from English to the eight African languages. Converting the English words to isiZulu and isiXhosa was easier as the researcher is familiar with the languages and also used the relevant dictionaries. In converting the English words to Setswana, the researcher sought help from his family members and a dictionary. The blank cells in Table 6.1 are words that the researcher could not obtain and therefore the Tshivenda and isiNdebele languages had the lowest number of created passwords for the top weak passwords as illustrated in Table 6.1.

Additionally as users use months as base words in their passwords, all the months were also converted to the eight languages selected (see Table 6.2). Each of the months was converted to its traditional name, because sometimes Zulu people call July *uJulayi*, which is different to the traditional word of *uNtulikazi*.

²<https://www.symantec.com/connect/blogs/top-500-worst-passwords-all-time>

Table 6.2: Months represented in the various languages

English	Zulu	Swati	Sepedi	Tshivenda	IsiNdebele	Setswana	Xitsonga	Xhosa
January	UMasingana	Bhimbidwane	Pherekong	Phando	Zibandela	Ferikgong	Sunguti	EyoMqungu
February	UNhloLANja	Indlovane	Dibokwana	Luhuhi	NhloLANja	Tlhakole	Nyenyenyana	EyoMdumba
March	UNdasa	Indlovu	Hlakola	Thafamuhwe	Mbimbitho	Mopitlwe	Nyenyankulu	EyoKwindla
April	UMbasa	Mabasa	Moranang	Lambamai	Mabasa	Moranang	Dzivamusoko	uTshaz'iimpuzi
May	UNhlaba	Inkhwekhweti	Mopitlo	Shundunthule	Nkwenkwezi	Motsheganong	Mudyaxihi	EyeCanzibe
June	UNhlangulana	Inhlaba	Phuphu	Fulwi	Nhlangula	Seetebosigo	Khotavuxika	EyeSilimela
July	UNtulikazi	Kholwane	Mosegamanye	Fulwana	Ntulikazi	Phukwi	Mawuwani	EyeKhala
August	UNCwaba	Ingi	Phato	Thangule	NCwabakazi	Phatwe	Mhawuri	EyeThupha
September	UMandulo	Inyoni	Lewedi	Khubvumedzi	Mpandula	Lwetse	Ndzhati	EyoMsintsi
October	UMfumu	Imphala	Diphalane	Tshimedzi	Mfumu	Diphalane	Nhlangula	EyeDwarha
November	ULwezi	Lweti	Dibatsela	Lara	Lwezi	Ngwanatsele	Hukuri	EyeNkanga
December	UZibandela	Ingongoni	Manthole	Nyendavhusiku	Mpalakazi	Sedimonthole	N'wendzambahala	EyoMnga

Section 2.2 reviewed publicly available password or end-user policies in Southern African organisations. It was found that most organisations require users to have a password that is at least six characters in length and contains a number and a special character. In Chapter 5 we analysed over 70750 user created passwords in South Africa and found that users created passwords that mainly followed the following hashcat mask conventions:

1. ?u?!?!?!?d?d?d?d – Uppercase *1, lowercase *3 and digits *4
2. ?u?!?!?!?!?d?d – Uppercase *1, lowercase *5 and digits *2
3. ?u?!?!?!?!?!?d?d – Uppercase *1, lowercase *6 and digits *2

With these hashcat masks and the African words, African derived passwords were created that correspond to the following mask:

1. ?u?!?!?!?!?!?d?d?s – Uppercase *1, lowercase *3, digits *2 and special character
2. ?u?!?!?!?!?!?d?d – Uppercase *1, lowercase *5 and digits *2

The key difference between the masks is that some of the passwords had a special character appended at the end. The months illustrated in Table 6.2 were appended with 18 at the end to signify 2018. Once the passwords were created they were hashed using NTLM and MD5 hashing algorithms. NTLM was first used and then followed by MD5 to attract more researchers attacking the password hashes. This was done because the researcher wanted to mimic an organisation that had users that used African word passwords hashed using the two algorithms.

6.2 Further processing of passwords generated

All the passwords generated were based on words in Tables 6.1 and 6.2 as stated and the Python script in Listing 6.1 was then run on these African words. The Python password generator takes the African words and appends a random number between 1 and 100 to each of them and a special character as well. These special characters are the special characters that were found and extracted from the cracked password list in Chapter 5. The generated African passwords were hashed to the NTLM format to mimic an organisation where the users create passwords in an African language and these are then stored in a text file.

Listing 6.1: Python password generator

```

#! C:\python27
import string, random, hashlib, binascii

def special():
    special_chars = ["!", "#", "$", "*", ".", "_", "%", "&", "+", "/", "-", "^", ") ",
                    "(", "?", "=", ";", ":", "]", "[", ">", "{", "}", "<", "\\ ", "'", "|", "~", ":"]
    abe = random.choice(special_chars)
    return abe

def number():
    number= random.randint(1,100)
    return number

text_file = open ("test.txt", "w")
with open('Words.txt') as words:
    for line in words:
        if len(line)>=6:
            #number= random.randint(1,100)
            capitalized_string = line.capitalize()
            full = capitalized_string+str(number()+str(special()+ "\n"))
            hash = hashlib.new('md4', full.encode('utf-16le')).digest()
            clear_hashed = hash + "," + full
            print binascii.hexlify(hash)
            text_file.write("%s \n" %clear_hashed)
            print full

text_file.close()

```

Statistics on the generated passwords are: 118 passwords had numbers and a special char-

acter, while 117 passwords had only numbers appended. All the African language months illustrated in Table 6.2 were appended with the number 18 at the end to mimic 2018 in short hand. Table 6.3 illustrates the generated password lengths of all the passwords, their counts and how often each password length appears in the generated list.

Table 6.3: Generated password lengths

Password length	Count n = 330	Frequency (%)
11	59	17.74
10	50	15.29
8	47	14.37
9	47	14.37
12	37	11.31
13	34	10.4
14	20	6.12
7	16	4.89
15	4	1.22
18	4	1.22
16	3	0.92
17	3	0.92
6	3	0.92
19	1	0.31
21	1	0.31
22	1	0.31

Passwords with a character length of 11 were the most frequent, followed by ten character passwords and in third place eight character passwords. The only restriction that was placed on the password length, was that the base word had to be six characters or longer to be used as a African language derived password. This was done because in all the organisational password policies that were reviewed in Section 2.2, the lowest password length was six characters and therefore it was best to implement the same restriction.

6.3 Analysis of African derived passwords

A total of 330 NTLM hashes that were created as described in Section 6.2, were uploaded to *Hashkiller*³ while ten random NTLM hashes were uploaded to *Onlinehashcrack*⁴. Between the two password cracking services *hashkiller* and *Onlinehashcrack*, 100 out of 330

³<https://hashkiller.co.uk/hash-list-manager.aspx>

⁴<https://www.onlinehashcrack.com/bf2b704bf8>

Table 6.4: Cracked words found per language

English	Zulu	Swati	Sepedi	Tshivenda	IsiNdebele	Setswana	Xitsonga	Xhosa
Password	iphasiwedi	Lekungena	Phasewete	Phasiwede	Ipasiwedi	Phasewete	Phasiwede	iphasiwedi
let me in	ngingemise					Mpolele		ndifake / phakathi
welcome	wamukelekile	Mukelekile	Amogetswe	Davhidza	Siyalemukela	O amogetswe	Amukela	Wamukelekile
Freedom	inkululeko		bolokolohi					inkululeko
hello	Sawubona	Sawubona	Thobela	Matsheloni Avhudi	Lotjhani	Dumela	Avuxeni	Molweni
football	ibhola	ibhola	bolo	bola		bolo	Bolo ya minenge	ibhola
master	inkosi	inkosi	Kgosi	Thovhela	Ikosi	Kgosi	Hosi	Ukumkani
bible	iBhayibheli	iBhayibheli	Bibele	Bivhili		bebele	Bibele	iBhayibhile
fish	inhlanzi		hlapi			tlhapi	Nhlampfi	intlanzi
july	uJulayi	Julayi				Phukwi	Mawuwani	EyeKhala
summer	ihlobo	Ihlobo	lehlabula			selemo	Ximumu	ehlobo
come in	Ngena	ngena				kena		Ngena
shadow	isithunzi	sifuntsi				seriti		isithunzi
monkey	inkawu	inkawu				tsweni		isilwanyana
hunter	umzingeli	liphisi						ngumzingeli
trust no one	ungathembemuntu					setshepi/ motho		suthemba namnye
access	ukungena							fi kelelo
hammer	isando	candvula						nyundo
secret	imfihlo					sephiri		mifihlelo

NtLM hashes were successfully cracked. The remaining 220 passwords were converted to MD5 and uploaded to *hashkiller*. An additional 38 more passwords were successfully cracked giving a grand total of 140 successfully compromised African passwords or 43%.

6.3.1 African language base words found

Understanding which base words were found can provide a better understanding of which words are used in researchers' wordlists. Table 6.4 shows the words that were found in the cracked password list. There are 97 words in this table and the words found in the cracked password list accounted for 65%. The words highlighted in pale orange are the words that were found by researchers while the words highlighted in burnt red are words where only one of the two words was found. The reason for this is that the direct translation of *ndifake* means "put me in" while *phakathi* means "inside". However people sometimes use these two words interchangeably and therefore both were tested. Table 6.4 shows that all the languages had words that were cracked. isiZulu and Setswana language words were the words that were found the most in Table 6.4, followed by isiXhosa words. The words that were found for all the languages are:

- hello
- July
- summer

Another interesting finding shown in Table 6.4 is that the word *password* was only found for isiZulu and isiXhosa represented as *iphasiwedi* for both languages.

Table 6.5: Analysis of words depicting months found

Months	Zulu	Swati	Sepedi	Tshivenda	IsiNdebele	Setswana	Xitsonga	Xhosa
January	UMasingana	Bhimbidwane	Pherekgong	Phando	Zibandela	Ferikgong	Sunguti	EyoMqungu
February	UNhloLANja	Indlovane	Dibokwana	Luhuhi	NhloLANja	Tlhakole	Nyenyenyana	EyoMdumba
March	UNdasa	Indlovu	Hlakola	Thafamuhwe	Mbimbitho	Mopitlwe	Nyenyankulu	EyoKwindla
April	UMbasa	Mabasa	Moranang	Lambamai	Mabasa	Moranang	Dzivamusoko	uTshaz'iimpuzi
May	UNhlaba	Inkhwekhweti	Mopitlo	Shundunthule	Nkwenkwezi	Motsheganong	Mudyaxihi	EyeCanzibe
June	UNhlangulana	Inhlaba	Phuphu	Fulwi	Nhlangula	Seetebosigo	Khotavuxika	EyeSilimela
July	UNtulikazi	Kholwane	Mosegamanye	Fulwana	Ntulikazi	Phukwi	Mawuwani	EyeKhala
August	UNcwaba	Ingi	Phato	Thangule	Ncwabakazi	Phatwe	Mhawuri	EyeThupha
September	UMandulo	Inyoni	Lewedi	Khubvumedzi	Mpandula	Lwetse	Ndzhati	EyoMsintsi
October	UMfumu	Imphala	Diphalane	Tshimedzi	Mfumu	Diphalane	Nhlangula	EyeDwarha
November	ULwezi	Lweti	Dibatsela	Lara	Lwezi	Ngwanatsele	Hukuri	EyeNkanga
December	UZibandela	Ingongoni	Manthole	Nyendavhusiku	Mpalakazi	Sedimonthole	N'wendzamhala	EyoMnga

Table 6.6: Cracked password lengths

Character Count	Count n = 139	Frequency (%)
8	38	27.34
9	28	20.14
10	22	15.83
7	16	11.51
11	15	10.79
12	11	7.91
13	5	3.6
14	3	2.16
6	1	0.72

Table 6.5 shows the words that were successfully cracked highlighted in pale orange. Of the 96 words only 57% of these were found. This is slightly better than the results in Table 6.4. The isiXhosa language only had two words for months found compared to Setswana which had all the words found except two. The month of February had the lowest count with only two hits compared to the month of October which had all its words found except that in Xhosa.

6.3.2 Passwords found

As stated 138 passwords were found and the lengths for the found passwords are shown in Table 6.6. Eight character passwords were the most often found in the cracked password list which is different to Table 6.3 which had 11 character passwords as the most frequently found passwords. Eight character passwords only ranked third in Table 6.3. The longest password length that was cracked was 14 characters.

Of the 118 generated passwords with numbers and special characters, 24 were successfully

cracked by researchers. This accounts for 20% of the passwords that contain special characters, but only 7% of the total number of cracked passwords.

One of the reasons why the base words were found is because Southern African languages use the QWERTY keyboard and the majority of the Southern African languages assessed do not use special letters like *â, á, ö and ï*. The only two Southern African language that use special characters are Afrikaans *ë, ô, é, à* and Tshivenda, which uses special characters such as those illustrated in Figure 6.1. The base words used as passwords would have been compromised at some point or researchers or their bots might have scraped websites for words that have been used and collected them over time. The researcher was able to download dictionary files for each of the eight languages and had a list of each of the words shown in Tables 6.1 and 6.2.

The analysis shows that converting the top weak English passwords into the various Southern African languages does not offer any further security for the user. Using months as passwords did not offer any additional security either as the majority of these words appear in password lists. However the Tshivenda language did have the lowest count of passwords being successfully cracked even after removing the special characters. The language therefore may provide security if used with the special characters.

6.4 Summary

In this chapter we discussed how weak English passwords were chosen and converted to eight different African languages. This chapter discussed the creation of the African language derived passwords using the organisational password policies discussed in Section 2.2. It also looked at the generated password lengths and the password creation process using a Python script.

Converting top weak English passwords to the eight African languages, did not prevent these passwords from being cracked and attacked by researchers on the Internet. isiZulu, Setswana and isiXhosa languages had the most found words when compared to the other African languages. Moreover, traditional words for months were also found by researchers on the Internet except for those in the isiXhosa and Xitsonga languages.

Chapter 7

Conclusion and future work

7.1 Summary

In this research we analysed the passwords that were used in corporate South Africa and found that generally the Microsoft AD password policy shaped users' password creation processes and policies. Use of the weak LM hashing algorithm was also prevalent in corporate South Africa. This is either because the organisations may not be aware that this hashing algorithm causes the passwords created to be weak or because the organisation is aware but needs to use the algorithm for legacy applications.

The base words of passwords that were found were all English based words which is understandable because the business language in South Africa is English and therefore one can assume that many passwords would be in English. The finding that the top base word is the word *password* is an indication that many South African users use weak passwords when generating passwords. This could be because of the organisational password policy that requires them to change their password every 30 days and therefore as a coping mechanism, users choose an easy base word and add in numbers and special characters to differentiate that word. Li *et al.* (2014) in their research found that the word "*password*" also featured in the top five passwords for English based users and appeared heavily in this research as well. As stated because of the regular password changes users employed copying techniques such as using company names as their passwords. These are words that the user sees often and will help them remember their passwords.

Organisations should be actively checking and blocking users from using know weak base words such as *password*, organisational names, organisational products and months. These

base words are easily guessed by malicious actors who can generate possible passwords that use those base words and try brute force their way into the organisation. Actively checking and blocking known weak passwords is supported by NIST in the NIST 800-63b. The issue for South African organisations is that most of the users of computer systems have a low rate of computer literacy and even lower cyber security training around password management.

Converting the top English weak passwords into eight Southern African languages showed that there are no additional benefits in doing so because most of the base words were already in researchers' password lists and adding in numbers and special characters did not increase the password security. The words that were chosen were words that are easily found online and therefore could have been scraped from various websites. Also for the Tshivenda language all the special letter characters were removed which could have made the words weaker and easier to be cracked by researchers.

7.2 Future work

As stated the Tshivenda languages is one of the few Southern African languages that has special letters and to generate passwords that employ these special letters and testing their effectiveness would be something to explore. Africa as a continent contains many different languages that are indigenous to the area. Generating passwords that are derived from Swahili, Yoruba, Igbo and Amharic and testing their effectiveness would give a greater understanding of which languages researchers would struggle to crack. Additionally testing popular African languages for each region, for example popular languages in Northern Africa, Eastern Africa, Southern Africa and Western Africa and testing which area had the lowest count of cracked passwords.

African regions that use different keyboards should also be assessed as the words that are used and their positioning might be different to those using the QWERTY keyboard.

References

- Adams, Anne, & Sasse, Martina Angela. 1999. Users are not the enemy. *Communications of the ACM*, **42**(12), 40–46.
- Ashwini, R, Birendra, J, & Gananand, K. 2013. Effect of grammar on security of long passwords. *Pages 317–324 of: Proceedings of the third ACM conference on data and application security and privacy*. CODASPY '13. New York, NY, USA: ACM.
- Blocki, J, Komanduri, S, Procaccia, A, & Sheffet, O. 2013. Optimizing Password Composition Policies. *Computing Research Repository*, **abs/1302.5101**.
- Bonneau, J, Herley, C, van Oorschot, P, & Stajano, F. 2015. Passwords and the Evolution of Imperfect Authentication. *Commun. ACM*, **58**(7), 78–87.
- CapeNature. 2007. Policy on Acceptable Use of Information Technology Information Systems. Online. Available from: <http://capenature.worldwidecreative.co.za/docs/418/CapeNature%20IT%20IS%20Policy%20April%202007.pdf>. (Accessed: 3/01/2018).
- Center for Internet Security. 2016. *CIS Microsoft Windows Server 2012 R2 Benchmark*. Technical report. Center for Internet Security.
- Cornell, University. 2004. *Frequency Table*. <http://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html>. (Accessed on 08/05/2018).
- David, Coursey. 2011. *25 "Worst Passwords" Of 2011 Revealed*. <https://www.forbes.com/sites/davidcoursey/2011/11/21/25-worst-passwords-of-2011-revealed/#692f109e9b5b>. (Accessed on 07/18/2018).
- Dell'Amico, Matteo, Michiardi, Pietro, & Roudier, Yves. 2010. Password Strength: An Empirical Analysis. *Pages 983–991 of: Proceedings of the 29th Conference on Information Communications*. INFOCOM'10. Piscataway, NJ, USA: IEEE Press.

- Friendman, B. 2014. *A study of South African computer users' password usage habits and attitude towards password security*. Masters thesis, Rhodes University.
- Gemalto. 2017. 2017 Poor Internal Security Practices Take a Toll. Online. Available from: <http://breachlevelindex.com/assets/Breach-Level-Index-Report-H1-2017-Gemalto.pdf>. (Accessed on 02/10/2018).
- Gibbs, S. 2016. *SS7 hack explained: what can you do about it?* The Guardian. (Accessed: 4 January 2018).
- Goodin, D. 2012. 25-GPU cluster cracks every standard Windows password in <6 hours. Ars Technica. Available from: <https://arstechnica.com/information-technology/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours/>. Accessed: 12 January 2018.
- Grassi, P, Fenton, J, Newton, E, Perlner, R, Regenscheid, A, Burr, W, Richer, J, Lefkowitz, N, Danker, J, Choong, Y, Greene, K, & Theofanos, M. 2017. *Digital identity guidelines: authentication and lifecycle management*. Technical report. National Institute of Standards and Technology.
- Guidorizzi, Richard P. 2013. Security: active authentication. *IT Professional*, **15**(4), 4–7.
- hashcat. *rule_based_attack [hashcat wiki]*. https://hashcat.net/wiki/doku.php?id=rule_based_attack. (Accessed on 06/21/2018).
- Inglesant, P., & Sasse, A. 2010. The True Cost of Unusable Password Policies: Password Use in the Wild. *Pages 383–392 of: Proceedings of the Special Interest Group on Computer–Human Interaction Conference on Human Factors in Computing Systems*. CHI '10. New York, NY, USA: Association for Computing Machinery.
- Johansson, J. *Windows Passwords: Everything you need to know*. <http://download.microsoft.com/download/f/4/a/f4a67fc8-c499-461d-a025-8155fb4f7a0f/Windows%20Passwords%20Master%201.5%20Handout%20-%20Jesper%20Johansson.ppt>. (Accessed on 04/04/2018).
- Komanduri, S, Shay, R, Kelley, P Gage, M, Michelle L., Bauer, L, Christin, N, Cranor, L Faith, & Egelman, S. 2011. Of Passwords and People: Measuring the Effect of Password-composition Policies. *Pages 2595–2604 of: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM CHI Conference on Human Factors in Computing Systems. New York, NY, USA: Association for Computing Machinery.

- Letsebe, Kgaogelo. 2017. Cyber insurance demand grows ITWeb. (Accessed on 03/22/2019).
- Li, Zhigong, Han, Weili, & Xu, Wenyuan. 2014. A Large-Scale Empirical Analysis of Chinese Web Passwords. *Pages 559–574 of: 23rd USENIX Security Symposium (USENIX Security 14)*. San Diego, CA: USENIX Association.
- Matt, M. 2015. *A practical guide to cracking password hashes*. <https://labs.mwrinfosecurity.com/blog/a-practical-guide-to-cracking-password-hashes>. (Accessed on 10/30/2017).
- Mavretich, R. 2012. Using the Center for Internet Security (CIS) benchmarks to support an information security management system. Online. Available from: <https://www.sans.org/reading-room/whitepapers/iso17799/center-internet-security-cis-benchmarks-support-information-security-manageme-34112>. (Accessed: 18 February 2018).
- Merrell, Shelly. *Base Words: Definition & Examples - Video & Lesson Transcript | Study.com*. <https://study.com/academy/lesson/base-words-definition-examples.html>. (Accessed on 07/19/2018).
- Microsoft. *Configuring Password Policies*. <https://technet.microsoft.com/en-us/library/dd277399.aspx>. (Accessed on 07/30/2018).
- Microsoft. *Selecting Secure Passwords*. <https://msdn.microsoft.com/en-us/library/cc875839.aspx#mainSection>. (Accessed on 04/15/2018).
- Microsoft. 2008. *Store password using reversible encryption for all users in the domain | Microsoft Docs*. [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc957013\(v=technet.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc957013(v=technet.10)). (Accessed on 06/12/2018).
- Microsoft. 2014. *How the Data Store Works: Active Directory*. [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc772829\(v=ws.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc772829(v=ws.10)). (Accessed on 03/23/2018).
- Microsoft. 2016. *Ntdsutil | Microsoft Docs*. [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/cc753343\(v=ws.11\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/cc753343(v=ws.11)). (Accessed on 04/22/2018).

- Microsoft. 2018a. *Lightweight Directory Access Protocol (Windows)*. [https://msdn.microsoft.com/en-us/library/aa367008\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/aa367008(v=vs.85).aspx). (Accessed on 04/04/2018).
- Microsoft. 2018b. *[MS-ADTS]: Password Modify Operations*. <https://msdn.microsoft.com/en-us/library/cc223247.aspx>. (Accessed on 03/31/2018).
- Microsoft. 2018c. *[MS-ADTS]: unicodePwd*. <https://msdn.microsoft.com/en-us/library/cc223248.aspx>. (Accessed on 04/04/2018).
- Morgan, S. 2015. Cybersecurity market reaches 75 billion dollars in 2015; expected to reach 170 Billion dollars by 2020. Forbes.com. Available from: <https://www.forbes.com/sites/stevemorgan/2015/12/20/cybersecurity-market-reaches-75-billion-in-2015-expected-to-reach-170-billion-by-2020/>. (Accessed on 2017-09-30).
- Mpowered. 2015. Password policy password construction & use guideline. Online. Available from: <http://mpowered.co.za/wp-content/uploads/2017/06/Password-Policy.pdf>. (Accessed: 3/01/2018).
- National Institute of Standards and Technology. 1999. *FIPS 46-3, Data Encryption Standard (DES) (withdrawn May 19, 2005)*. <https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf>. (Accessed on 03/20/2019).
- Olivia, Waxman. 2012. *The 25 worst passwords of 2012 - CNN*. <https://edition.cnn.com/2012/10/25/tech/web/worst-passwords-2012/index.html>. (Accessed on 07/18/2018).
- Parkin, S, Driss, S, Krol, K, & Sasse, M Angela. 2016. Assessing the User Experience of Password Reset Policies in a University. *Pages 21–38 of: Technology and Practice of Passwords*. Springer International Publishing.
- Rahman, H. 2014. *Active Directory files and their functions*. <https://blogs.msdn.microsoft.com/servergeeks/2014/10/14/active-directory-files-and-their-functions/>. (Accessed on 03/23/2018).
- Rankin, K. 2012. Hack and password cracking with GPUs, Part II: get cracking. *Linux Journal*, **2012**(214).
- Republic of South Africa. 2011. *The Constitution of the Republic of South Africa, 1996 : as adopted on 8 May 1996 and amended on 11 October 1996*. Constitutional Assembly.

- SANS Institute. 2014. *Password Protection Policy*. <https://www.sans.org/security-resources/policies/general/pdf/password-protection-policy>. (Accessed on 01/04/2018).
- Sasse, M A, Brostoff, S, & Weirich, D. 2001. *BT Technology Journal*, **19**(3), 122–131.
- Savill, John. 2000. *What is a SID (Security ID)? | IT Pro*. <http://www.itprotoday.com/management-mobility/what-sid-security-id>. (Accessed on 06/10/2018).
- Schoeman, Willem J. 2017. South African religious demography: The 2013 General Household Survey. *HTS Teologiese Studies / Theological Studies*, **73**(2).
- Steube, Jens. 2018. *hashcat/kwprocessor: Advanced keyboard-walk generator with configurable basechars, keymap and routes*. <https://github.com/hashcat/kwprocessor>. (Accessed on 07/21/2018).
- Stuttard, D., & Pinto, M. 2011. *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws*. Wiley.
- Summers, W., & Bosworth, E. 2004. Password Policy: The Good, the Bad, and the Ugly. *Pages 1–6 of: Proceedings of the Winter International Symposium on Information and Communication Technologies*. WISICT '04. Trinity College Dublin.
- Symantec. 2010. *The Top 500 Worst Passwords of All Time | Symantec Connect Community*. <https://www.symantec.com/connect/blogs/top-500-worst-passwords-all-time>. (Accessed on 08/02/2018).
- University, Rhodes. 2016. Password guideline. Online. Available from: <https://www.ru.ac.za/media/rhodesuniversity/content/informationtechnology/Password%20Guidelines.pdf>. Accessed: 4/01/2018.
- Van Allen, F. 2013. *Google reveals the 10 worst password ideas*. <http://techland.time.com/2013/08/08/google-reveals-the-10-worst-password-ideas/?iid=biz-article-mostpop2>. (Accessed on 09/30/2017).
- Whitaker, A., & Newman, D.P. 2005. *Penetration Testing and Network Defense*. Networking Technology. Pearson Education.
- Whittaker, Z. 2015. Hacking Team used shockingly bad passwords. Online. Available from: <http://www.zdnet.com/article/no-wonder-hacking-team-got-hacked/>. (Accessed on 03/23/2018).