Bachelor's thesis

Degree Programme in Information Technology

Network Technology

2015

Darren Richardson

INFORMATION SECURITY

– An investigation into password habits

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Information Security

May 2015 | 37 pages

Instructor: Tero Virtanen

Darren Richardson

INFORMATION SECURITY – AN INVESTIGATION INTO PASSWORD HABITS

This thesis considers password security guidelines used in current environments and stipulates that password requirements force users to create and use passwords which are easy for computers to guess but hard for humans to remember.

The thesis begins by exploring a number of the most prevalent methods of illicitly obtaining passwords in an attempt to design an experimental method to test the notion of weak password distribution. Password cracking techniques are discussed, as well as less technical methods. Then the topic shifts into password security research, attempting to learn information about password security guidelines, entropy and length in relation to a password's level of security and, finally, the dangers of password reuse in a society where mass information systems are linked. Before the experiment begins, the author examines a number of passwords obtained from a social networking site to show how many flawed passwords are allowed by even today's advanced systems.

An experiment was then planned, in which a number of password cracking methods were used against a server in order to determine how large a threat password cracking is to a typical user. Monitoring software was used to watch network traffic in order to discover whether such traffic is obvious.

The results of the experiment show that password cracking techniques and technology have evolved far faster than password security guidelines and that entropy in the form of length is the strongest method of securing passwords. The thesis gives a number of updated password guidelines in order to prevent access via illicit means.

KEYWORDS:

Passwords, Security, Social Media, Penetration Testing, Networking.

CONTENTS

TABLE OF CONTENTS

FIGURES

TABLES

LIST OF ABBREVIATIONS (OR) SYMBOLS

Entropy	Entropy, in information theory terms, generally means disorder or uncertainty. As an example we can use the Bernoulli process in checking a coin toss results. If the coin is fair (has an equal chance of coming up heads and tails) then it has the maximum possible entropy because it is the most difficult to predict the next outcome. If the coin tends towards heads (i.e. It is not a fair coin) then the entropy reduces.
CPU	Central Processing Unit - Central Processing Units are problematic when it comes to rapid calculation. This is because a CPU has 2-8 cores to be used in calculation while GPUs contain hundreds that can be used in rapid calculations in parallel
GPU	Graphics Processing Unit - GPUs are the main component of graphics cards used in personal computers. Due to the highly parallel structure they are far more effective than CPUs when it comes to rapid calculation. For this reason they are used in calculation systems such as password cracking and bitcoin mining.
SQL	Structured Query Language is a language used to manipulate and manage data stored in a relational database management system.
MySQL	MySQL is one of the most used relational database management system.
ARP spoofing	ARP spoofing is the act of sending false (or spoofed) ARP packets onto a network with the intention of confusing the ARP process.
SQL Injection	An attack in which partial SQL commands are injected using database entry forms.
Hash	A hash is a mathematical function that is practically impossible to invert, and thus used to store sensitive information.
Salting	The act of adding random bits to stored password databases in order to prevent dictionary attacks or rainbow tables.
SHA-1	Secure Hash Algorithm, a commonly used hashing function.
TCP	Transmission Control Protocol, a core protocol within the Internet Protocol Suite. It is used to provide structured, error checked and reliable connections.
ARP	Address Resolution Protocol is used for the resolution of network layer addresses into link layer addresses.

Loopback	A loopback address is used to route an outgoing signal directly back to the sender, usually used to diagnose connection issues.
Nmap	A network mapping tool.
Metasploit Framework	A complete framework for building, transmitting and executing exploits on remote systems.
Meterpreter	An in-built command line tool within Metasploit Framework, giving access to a number of root level tools such as dumping encrypted storage.
xHydra	A tool used to execute dictionary attacks against online services such as SSH.
John the Ripper	A tool used to execute dictionary attacks against offline resources, such as Windows .sam files and Linux passwd/shadow files.
Rainbow Tables	Tables containing precalculated hashes used to attack weak hashes.

1 INTRODUCTION

Passwords play a significant role in the life of the average user. As the primary form of gaining access to accounts and services the function of the password faces heavy scrutiny from the IT community. While arguably more secure systems exist, such as biometrics or one-time passwords, user-defined passwords continue to have a nigh-universal role in IT infrastructure. It is not possible for a person to manage, either personally or professionally, without the use of user-defined passwords.

Passwords, along with the associated account names, are a way to indicate who we are online. They are used to control social media, email, banking and a plethora of other services. As such, user accounts can be seen as our personal presence on the internet. As with any method of personal identification it is vitally important to keep our user accounts safe. A malicious agent obtaining a username and password is akin to a fraudster gaining access to a person's banking details and home address. A great deal of damage can be done with even the smallest of information.

A great deal of password alternatives exist. Hardware verification, such as security tokens are often used in corporate situations, particularly for delicate and/or restricted network access. The cost of such systems prevent wide spread distribution, however, given that an authority capable of the maintenance and issuing of such tokens is required. Biometric evaluation, such as finger print scanning, provides another more secure form of access but in several situations such measures are impractical due to expense of implementation.

This thesis sets out to discuss the idea that despite the prevalence of the user/password identification system, people rarely use passwords of any reliable strength or complexity, instead sticking rigidly to out-dated guidelines which make passwords difficult for people to remember but easy for computers to break.

2 SECURITY RISKS AND RESEARCH

Before a testing system can be designed, we must first stipulate what sort of testing is required to support the idea that passwords are easy for computers to guess. The research is divided into two parts. The first part will discuss ways to gain access, either by cracking a password or other (technical and non-technical) means. The second portion will explore the password use of general users, and the guidelines that dictate their choice of passwords.

2.1 Methods of cracking

A great deal of methods exist for gaining access to systems by bypassing the standard security settings. These methods may range from executing small portions of code using exploits on the vulnerable machine to gain complete control via backdoors or providing illicitly obtained but legitimate login information.

This thesis focuses on the latter, though options about the former will be discussed to provide a base for comparison. We will touch on the following areas, although it is by far not a comprehensive list:

- Phishing and Keylogging
- Man-in-the-Middle attacks

We will also research a number of options regarding password security and cracking methodology:

- Dictionary attacks
- Brute Force attacks
- Online vs Offline attacks

2.1.1 Phishing/Keyloggers

A number of attack vectors exist that require little or no technical knowledge of security practices and protocols. While the use of these attack vectors is beyond the scope of this thesis, their impact on the world of information security mandates that we discuss them in, at least, a basic form.

Microsoft security pages state that "Social engineering is a technique used by criminals to gain access to your computer. The purpose of social engineering is usually to secretly install spyware or other malicious software or to trick you into handing over your passwords or other sensitive financial and personal information." [1]

This practice, often known as Phishing, follows several different methods. A common technique heavily used in the UK between 2005 and 2008 was known as Phone Phishing where a group or individual would contact a person via phone and claim to be a trusted company such as a bank. They would go on to claim they had found problems with a bank account and ask for various details.

More technical Phishing attacks may include imitation of official communication, again from a bank or other trusted source asking for usernames, passwords or credit card details.

Link manipulation is a common technique used in these forms of attack. For example, the attackers may use sub-domains or a misspelled URL in order to make it appear as though communication is official.

The attackers may even go so far as to forge a copy of the companies website or using cross site scripting direct a target to sign in using their own bank service where everything may appear to be correct. One such attack was used in 2006 to attack the Paypal service.

Keyloggers pose a different threat altogether. A keylogger is a piece of software designed to be installed on a computer through illicit means. Either a backdoor or a trojan horse would be a perfectly suitable delivery method though many others exist. This program is usually set to execute silently on startup and capture every keystroke made by the target machine.

An example of such a program is inputlog though this software is used for research purposes. Examining data from a keylogger can provide user-names, passwords, access to sensitive correspondence and other breaches of confidentiality and security.

Hardware keyloggers also exist, acting as a bridge between the keyboard and computer. These are practically undetectable by software prevention methods though they have an obvious increased risk of discovery.

2.1.2 Man-in-the-Middle attacks

Man-in-the-Middle attacks are an attack vector in which traffic going between two hosts is intercepted. Imagine a situation in which a user's computer, hereafter Host-PC connects to the server, hereafter Server-A. The setup is demonstrated in Figure 1.

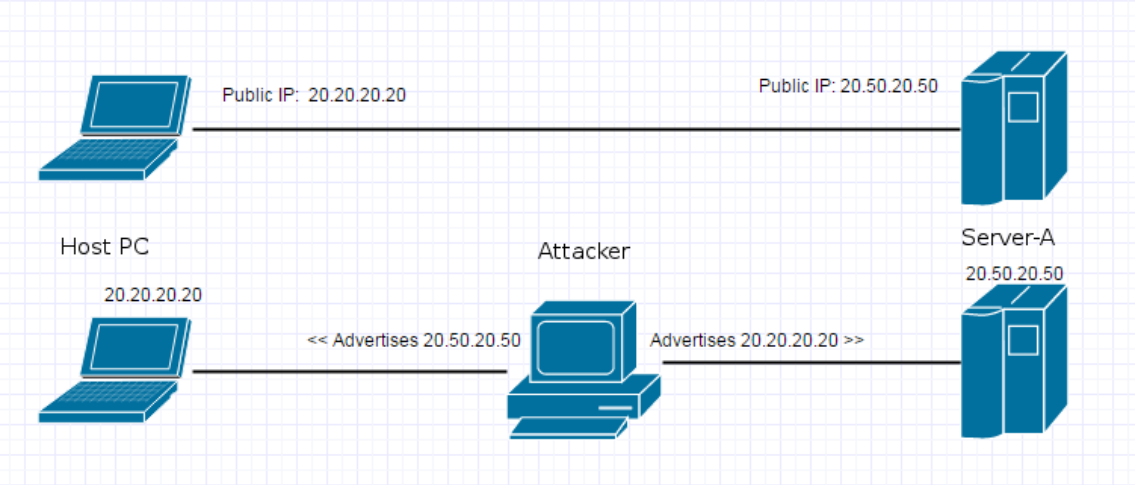

Figure 1. Man-in-the-Middle attack setup

Host-PC connects to Server-A and Server-A requests login information. In a normal situation, this information goes from Host-PC to Server-A without incident. During a Man-in-the-Middle attack (displayed in Figure 1), the attacker (Attacker) positions himself between the Host-PC and Server-A with the intent of maliciously 'sniffing' network traffic. There are several methods of doing such, but for this example, we will assume ARP spoofing has been used.

Attacker sends spoofed ARP packets to the Host-PC, claiming to be Server-A. Similarly, it sends spoofed ARP packets to Server-A, this time claiming to be Host-PC. As such, traffic destined for Host-PC from Server-A is instead sent to Attacker, as is all traffic meant for Server-A from Host-PC. In this situation, when Host-PC sends the login details 'User=Host Password=applesux' the information goes through Attacker-PC.

Depending on the attackers motives several situations can occur. A Denial-of-Service attack can be implemented at this point, by simply dropping all the associated network packets. This can be used to give the impression that Server-A's services are down, especially if other network traffic is forwarded as normal.

Alternatively, if the attacker is sniffing packets 'passively', he can observe traffic as it passes. In this situation, provided there is no encryption on the traffic, he would see the username and password as it passes over his connection before the packets are forwarded as normal.

In an active setting, the attacker might want to modify traffic as it goes. For example, after the legitimate login attempt, User-PC might send a 'message' saying “I am home at 16 today.” The attacker might alter this to say “I am home at 14” for example.

An article on lifehacker also notes that “A man-in-the-middle attack can be successful only when the attacker forms a mutual authentication between two parties. Most cryptographic protocols always provide some form of endpoint authentication, specifically to block MITM attacks on users.” [2]

While Man-in-the-Middle attacks are an interesting situation to consider and have a great deal of literature devoted to them, this thesis is not one of them. As such, Man-in-the-Middle attacks will be discussed only theoretically.

2.1.3 Brute Force attacks

Brute force attacks rely on being both a simple and comprehensive attack. Instead of exploiting vulnerabilities in software or networking systems, brute force attempts to make legitimate login attempts until one is accepted as correct

and access is gained. Initial research into a system during penetration testing can often provide a list of usernames, which can be used in conjunction with these attacks.

Such attacks work by literally testing every keyspace for each possible value, attempting upwards of millions of tries a second until the password is broken.

Attacks such as this leave obvious traces. Server memory usage may bloat because of having to serve such a large number of login requests. Network traffic will certainly be very obvious with so many login attempts.

As such, Brute Force attacks are better off running offline.

Brute force attacks rely heavily on hardware. Like Bitcoin mining, Brute Force attacks require high numbers of parallel calculations made. Therefore, processes that run on GPUs rather than CPU-based calculation are far faster. Later we will discuss a machine created to execute Brute Force attacks.

2.1.4 Dictionary attacks

Dictionary attacks are a compromise to using a full brute force approach. Instead of trying every possible keyspace, a 'dictionary' file is used as the basis for cracking. A dictionary file is a plaintext file containing passwords separated by lines. Dictionary files can be many megabytes in size, holding hundreds of thousands of passwords.

Hardware makes far less difference in Dictionary attacks. No calculation is required. Instead read speed is the deciding factor, but with little information to read it mostly relies on network traffic. Obvious attempts are still easy to stop with a dictionary attack.

The trade-off for the increased speed is that only common passwords are able to be cracked by dictionary attacks.

Some programs allow for 'mask' attacks which apply common replacements to letters and numbers in order to create a hybrid between brute force and dictionary attacks. This possibly provides the best compromise between efficiency of speed and the reliability of results.

2.1.5 Online vs Offline attacks

Online attacks refer to attacks launched using existing network connections. Attacks such as these have obvious drawbacks. Online attacks will consistently challenge password-protected systems with login attempts. As such each attempt will be treated as legitimate. Each attempt will display normal network traffic. Normal attempts at login might be considered as 1 to 4 tries. Anything higher raises a red flag, and online attacks rely on the ability to run make between tens of thousands and millions of attempts.

Another concern is that online attacks are subject to a great deal of security measures. IP blacklist software can be used to stop connection attempts from malicious hosts, who can be dicerned from an abnormal number of connection requests.

Furthermore, network bandwidth could be a limiting factor. While a dedicated password-cracking machine could make millions of attempts a second, network traffic restrictions might limit that number to thousands, perhaps even hundreds.

Offline attacks are much more powerful. When a machine can be dedicated to cracking a stored hash, the speeds attained are far higher than the online counterpart. The great disadvantage of offline cracking is in that a password file needs to be locally stored, and the logistics of obtaining such a file implies that the attacker already has a level of privileged access to the remote server he is attempting to penetrate.

This is not always the case. Some networks may have a centralized password server, and as such this would be a high priority goal during a penetration test.

If one is able to obtain the password file, an offline attack will be far more powerful.

2.2 Password security research

The next step in the research process is to move onto the area of password security, both theoretical and practical. The area of password security is such an expansive field that much of it lies beyond the scope of this thesis. Here we will research the following areas:

- Password Security Guidelines
- Length / Entropy
- Reuse
- Examples from website X

2.2.1 Password security guidelines

Password security and the role it plays in the professional and personal lives of the public has always been a source of concern and controversy. Symantec, a prominent IT security company, outlines[3] the points of a good password as:

- No personal information
- No dictionary words, proper nouns or foreign language words
- Password length between 6 and 9 characters
- Uses all possible character sets. Uppercase, Lowercase, Numbers, Symbols.

The summary of the thesis uses the standard IT theory that a good password is easy to remember, but hard to guess. They also warn against bad practices, such as writing down or sharing passwords.

The statement from Symantec in question was written is 2002. Similarly, the support pages for Google states that passwords require a minimum of 8 letters. This shows that over the last 13 years, while password cracking techniques and machines have been improving and, behind the scenes, password storage hashes have also been improving, users have no additional responsibility.

This stagnation in user-end security policies is echoed across most of the popular Internet services. Facebook's support page states that the service requires a minimum of 6 letters[4]. This means that one of the most popular services in the world has the same password security requirements as a user from 2002. Neither service, Gmail or Facebook, mandates a 90-day period between password changes as is a standard of good practice in IT security.

2.2.2 Length / Entropy

“Effective evaluation of password strength requires a proper metric. One possible metric is information entropy” has been stated in a paper presented to the 2012 IEEE Symposium on Security and Privacy[5].

Eight characters is the minimum length for a password to be secure if it takes advantage of all the potential character types. With 26 possibilities from lower case, 26 from uppercase, 10 from numbers and 12 from the full set of symbols this means every keyspace has 74 possible entries.

Note: ` ~ ! @ # $ % ^ & * ( ) _ - + = { } [ ] \ | : ; " ' < > , . ? / A number of these cannot be used with password systems. A total of 32 symbols are present. However given the prevalence of disallowing symbols which could cause issues such as SQL injection, we can use 12 as the number of available symbols.

From this we can determine that the possible combinations for an 8-letter password is 74^8. This means there are just under 900 trillion combinations of passwords. At 2 million guesses per second this would still take 14 years to complete.

However, at the Passwords^12 Security Conference in December of 2012 Jeremi M. Gosney gave a presentation that stipulated password crackers need more power. He went on to introduce a system that contained 25 AMD Radeon GPUs[6].

Ignoring bandwidth constraints, against the LM Hash, this system was capable of putting out 20,000,000,000 password attempts a second. Using this system on the 8-character password with 74 possible entries would lead to the password being broken in just over 12 hours.

Limiting oneself to only a subset of the available symbols can dramatically reduce the strength of the password. Using the alphanumeric subset of 62 characters (with a speed of 2 million attempts per second), the crack would take three years to complete. Using only lowercase and numbers the subset is only 32 characters and would complete within 6 days. Against a more powerful system, such passwords would hold up for mere seconds.

Length can serve to offset this issue somewhat. Each additional character space that needs to be calculated increases the entropy of a password. Even the addition of a single extra letter gives an additional exponent equal to the number of characters in the subset used.

For example: Increasing the 32-character subset's password length to 10 instead of 8 increases the cracking time (at 2 million guesses per second) to over 17 years.

When we increase the length of a password using 74 possible characters to 12, we see such an exponential increase in the cracking time that even Jeremi Gosney's system would need 4750 years to break the password when working against MD5 hashes.

From this, it may be possible to conclude that password length rather than entropy of the character space subset used is a more important factor in password security. We will aim to test this in the thesis.

2.2.3 Reuse

Given network restrictions on attacks password reuse is likely a more serious issue than password-cracking. Take the following example.

A user with the ID of 10T uses a password for his Gmail account that is very secure, 19 alphanumeric/symbolic characters. For all intents and purposes this password is uncrackable. However using a keylogger an enterprising hacker has obtained this information. He finds that the Gmail account is merely for collecting spam from other websites. Browsing through the emails can offer up usernames for such websites.

When this 19-character password is reused, the purpose of high entropic levels is rendered moot. Our enterprising hacker has obtained access to all of User ID 10T's information, including an official email address. Our enterprising hacker then creates an attack using various potential username schemes and password variants. Using this one password, this attacker has gained access to a company's secure network.

Research conducted at Microsoft[7] has some interesting results to share on this subject. “Among our interesting findings is how large a role web passwords play in users lives. The average user has 6.5 passwords, each of which is shared across 3.9 different sites. Each user has about 25 accounts that require passwords, and types an average of 8 passwords per day.”Examples from website X

On June 6, 2012 there was a dump of passwords from a well-known professional networking site containing 24132 passwords. The Python script in Table 1 was made to take an average of a selection of passwords.

Table 1. Line length averaging script written in Python

```
total = 0

count = 0

file = open('passwords', 'r')

for line in file:

        print(line)

        stringlength= line.__len__() - 1

        total = stringlength + total

        print stringlength

        print(total)

        count = count + 1

        print(count)

        print

        print

average = float(total) / float(count)

print average
```

First, we took a random sample of 200 passwords, finding them to be 9.095 characters in length. Then we tested the script against the entire file and came up to 9.279 characters long. While on initial look, these lengths make the

passwords seem strong, looking at the three samples from those provided in Table 2 reveals otherwise.

Table 2. Three password samples

jjmsjjm	**breaksbreaks**	**shallyqian**
J0n0th0n	**20FA89**	Go4number1
Lausebengel	M1ll1ona1re	Shoey2
Poeldijk1	Chingerel0	Ca4350799
8fmabg7k	Bigtoby1	Hels1nki
Komigjen1	**class_a**	Ca55anova
sunyingjie	280275s10810	Work4kids
Leshen7	Houghto1	Shanesmith
Our2boys	Guldsko	
15MAI1984	Colepaige1	
ANNAEMILIA	choco24nuts	
nmstnmst	not4everyone	

From the 32 passwords listed several are emboldened. These represent remarkably weak passwords to have been accepted on such a website. They are either examples of dangerously small passwords or passwords containing a single subset of characters from the possibilities.

Three passwords have been indented. These passwords, 280275s10810, 15MAI1984 and Hels1nki, initially seem strong. However, let us look more closely at these individually.

280275s10810 - The first six characters of this password are clearly a date of birth. The possibility exists, with mask attacks, to crack this password very simply by applying the following rules:

The first six numbers are a date of birth. Therefore each has a potential keyspace of 10.

The last six are alphanumeric with no upper case.

Therefore:

(10^6)+(32^6) == 1074741824 potential passwords. Our 2-million-guesses-a-second system would have this cracked in just 9 minutes. This is the reason using a date of birth in a password is bad practice.

Hels1nki - This password reveals a lot about the user. A person using this password is most likely Finnish and more often than not living in Helsinki. Replacing a I for a 1 and capitalization of the first letter is very common practice in passwords. We can stipulate the following:

(72^1)+(32^7) == 34359738440 passwords. Our system would break this password in just less than 5 hours.

15MAI1984 also reveals a great deal about the user. Obviously, a date of birth, the spelling of the month reveals that the user is most likely French. Just as a username should give no hints to a password, the password should similarly give no hints to the user.

Further examples of weak passwords in this way can be seen from the list as well in the example of Work4kids, Shanesmith (which is potentially just a name) and many others. Browsing through the passwords demonstrates that almost all of them have a weakness in this way, which probably contributed to the fact that they were cracked when leaked. Further contributing to the fact was the way they were hashed in unsalted SHA-1.

Some cryptographic hashes make use of 'salting'. In cryptography, a salt is random data added to a password hashing function with the intent to create extra keyspaces and/or entropy within the password. It is important to note that, when using a dictionary attack against a single password, salting will have no impact. However, it does make it more difficult to crack a list of passwords. Take the following example:

When cracking a password list, an attacker would compute a hash attempt (hereafter known as Attempt-1.) The chance of Attempt-1 coming up with a

match increase for the number of passwords in the file. However (if using early Unix implementation of 12-bit salt containing 4096 possible salt values), then an attacker has to process Salt-1-through-4096 for each attempt. Attempt-1+Salt1, Attempt-1+Salt2, Attempt-1+Salt3 through to Attempt-1+Salt4096.

Thus salting prevents the 'reusing' of hashes when attempting to break large lists of passwords.

Given that Rainbow Tables use large lists of pre-computed hashes for common passwords, salting also poses an obstacle for their use.

3 PASSWORD STRENGTH TESTING

The testing phase aims to prove that password security requirements as largely inaccurate or obsolete. A testing system will be setup wherein two computers are connected to a network and a number of attack attempts will be launched.

Two testing systems were initially considered. The first system connected a server to the attacking machine directly, each NIC card cabled to each other and an address scheme designated. This setup would maintain the purity of the experiment, given the absence of outside traffic. However, it was decided that this system's isolated nature would provide improper results and that a new system would be designed with maintaining regular traffic.

This second proposed system added both the server and attacker machine to an already existing, populated network with light traffic. Such a setup removes the purity of the experiment but provides additional relevance to a real-world setup. Interpretation of the network traffic is likely to be a large part of the results of such a test and, as such, keeping the experiment as close to real-world scenarios as possible.

Monitoring software for the network would have to be used. This is due to the fact that an obvious attack is likely to be noticed by even the most basic of Intrusion Prevention Systems such as those found in home routers. The monitoring software watched for changes in network traffic, as well as provide examples of genuine connection attempts for comparison.

A selection of programs were used to provide different sources and attacks. A dictionary attack uses the provided wordlist to attack systems. Brute force attacks will be demonstrated also to a point. However, due to the time requirements of in-depth brute force attacks they can only be theoretically discussed.

In order to prove that password security requirements have advanced at a far slower rate than password cracking technology and techniques, several factors need to be monitored during the testing phase.

While offline attacks of a captured hash are out of the scope of this thesis, examples of the speed and detectability of such methods will be discussed for the comparison between online and offline attacks.

3.1.1 Testing System

The testing system was attached to a network in order to ensure accurate conditions as close to real-world as possible. Two computers were used based on the specifications provided in Table 3.

Table 3. Testing computer specifications

	Attacker PC	Server PC
CPU	Pentium D	Pentium D
RAM	2GB	1GB
HDD	160GB	80GB / 160GB
NIC speed	100mbps	100mbps
IP Address	192.168.1.83	192.168.1.85
GPU	Radeon X550	N/A
OS	Kali Linux	Debian (Virtual Machine)

The addition of a graphics card in Attacker PC may seem arbitrary but, as brute force attacks require parallel calculation, the GPU is far better than the CPU. Given the exponential increase in processing power of GPUs since the release of the Radeon X550, it is safe to assume any parallel calculations will be faster under today's systems.

The systems were appended to an existing network as documented in Figure 2.

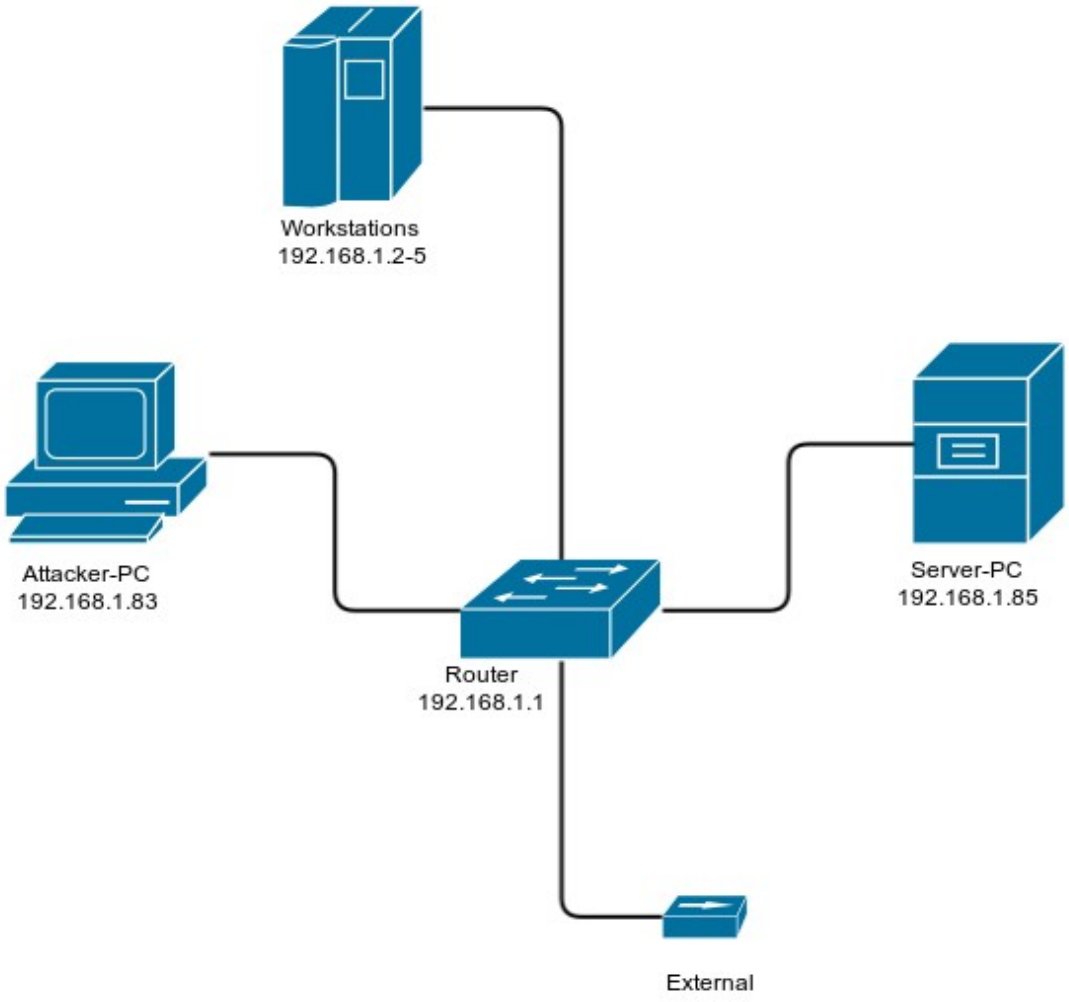

Figure 2. Documentation of the testing network

The Standards of Good Practice for IT Security[8] states that ”Network activity should be monitored using a range of techniques” in order to ”Assess the performance of the network, reduce the likelihood of network overload and detect potential or actual malicious intrusions.” As such, the connections between the Attacker-PC at 192.168.1.83 and the Server-PC at 192.168.1.85 would be monitored using Wireshark.

The Server-PC would attempt to simulate an access server in a working environment. It would have minimal security features (given that this thesis discusses security of passwords, rather than preventitive measures.) A number

of user accounts were created, each with incrementally secure passwords and higher permissions depending on their role inside the virtual environment. They were assigned the attributes outlined in Table 4.

Table 4. User and password setup values

Name	User Name	Role	Password	Strength
Thomas Jensen	Tjensen	Head of IT	M3OinPiG-WttfN	Strong
Alex Donald	adonald	IT department	There1sN0Spo0n	Fairly Strong
Suki Richards	Srichards	IT intern	Weakp1	Weak
Claire Porter	Cporter	CEO	MediuMPa	Medium
Jean Russell	Jrussel	Accounting	weakpa	Very weak

3.1.2 Test Log

In order to maintain some resemblence of authenticity with real world conditions the testing was divided into four stages. Information gathering, target probing, penetration and thesisation.

”Network discovery uses a number of methods to discover active and responding hosts on a network, identify weaknesses, and learn how the network operates. Both passive (examination) and active (testing) techniques exist for discovering devices on a network. Passive techniques use a network sniffer to monitor network traffic and record the IP addresses of the active hosts, and can report which ports are in use and which operating systems have been discovered on the network.”[9]

3.1.3 Information Gathering

The information gathering stage began by analyzing the network. To do this we used the program NMAP to do a primary ping sweep of the network using the following command:

```
nmap -sP 192.168.1.1-254
```

This allows us to get a general layout of the network and confirm that the hosts are set up as in our network diagram. Attacker-PC pings at 192.168.1.83 and the loopback interface on 127.0.0.1. Server-PC accepts ping requests on 192.168.1.85. The network was functioning as expected.

We proceed with the information gathering phase by running a comprehensive scan in NMAP. We begin with a TCP connect scan. This scan works by attempting to initiate the standard networking three way handshake. It is a graceful attack, in that it opens and closes connections properly and avoids the risk of flooding the server with connection requests. The command used is the following:

```
nmap -sT -p- -pN 192.168.1.85
```

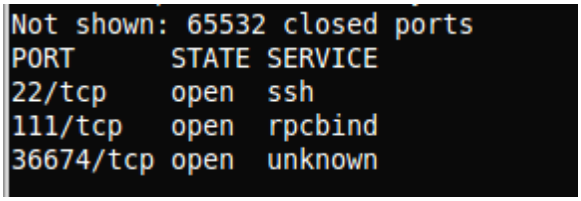

Figure 3. Nmap command output

Figure 3 shows that the SSH system is open and on a predictable port. In a business environment there would be other scans to run. For example, a more stable scan might be considered, for example the SYN scan. It is faster than a TCP scan because SYN scans fail to complete all three steps of the 3-way handshake used in TCP connections. Another consideration might be to check for open UDP ports. One might even consider sending more agressive scanning attacks. However, for the purpose of this thesis, the TCP information gathering phase is suitable.

3.1.4 Online Attack Phase

Before we continued on to online attacks, we started Wireshark running on the Server-PC in order to view network traffic. We do this in hope of noticing the difference between standard network chatter, legitimate traffic and erroneous traffic that is likely an attack. We then use xHydra to attempt a dictionary attack. As thesised by Figure 4, we target the Server-PC on the SSH protocol running on Port 22. We will specify a username with a weak password (JRussell) and run a small password file through to check the network traffic.

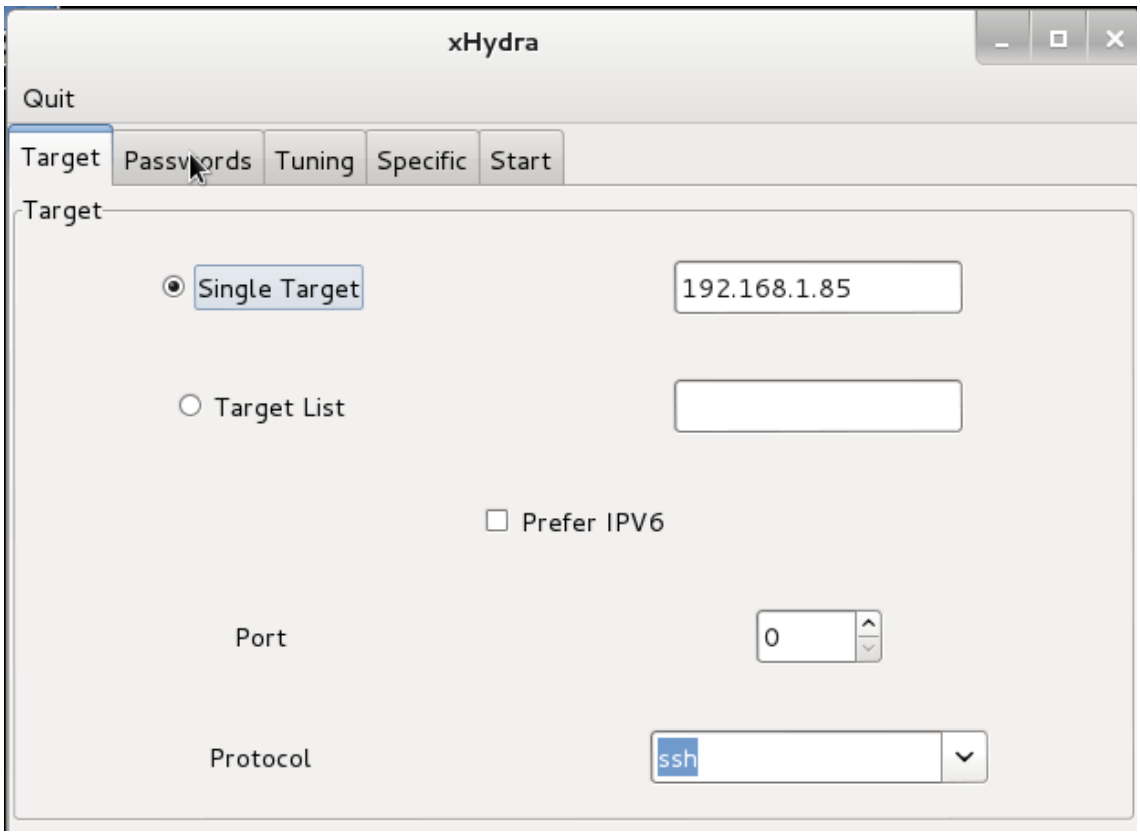

Figure 4. xHydra dictionary attack setup

A dictionary file must be specified in order for xHydra to run. The dictionary file in use is over 3 megabytes in size and contains over 300,000 passwords. There has always been something of a trade-off when it comes to dictionary files. Too large a file makes for slow cracking. Some dictionary files exist in excess of 1 gigabyte in size. However too small a file can severely limit the number systems

that will be vulnerable to the file.The password-cracking program, xHydra, runs at approximately 180 tries a minute, three password attempts a second. This speed is dramatically lower than our earlier estimates and will take many hours to run through even a small scale dictionary attack.

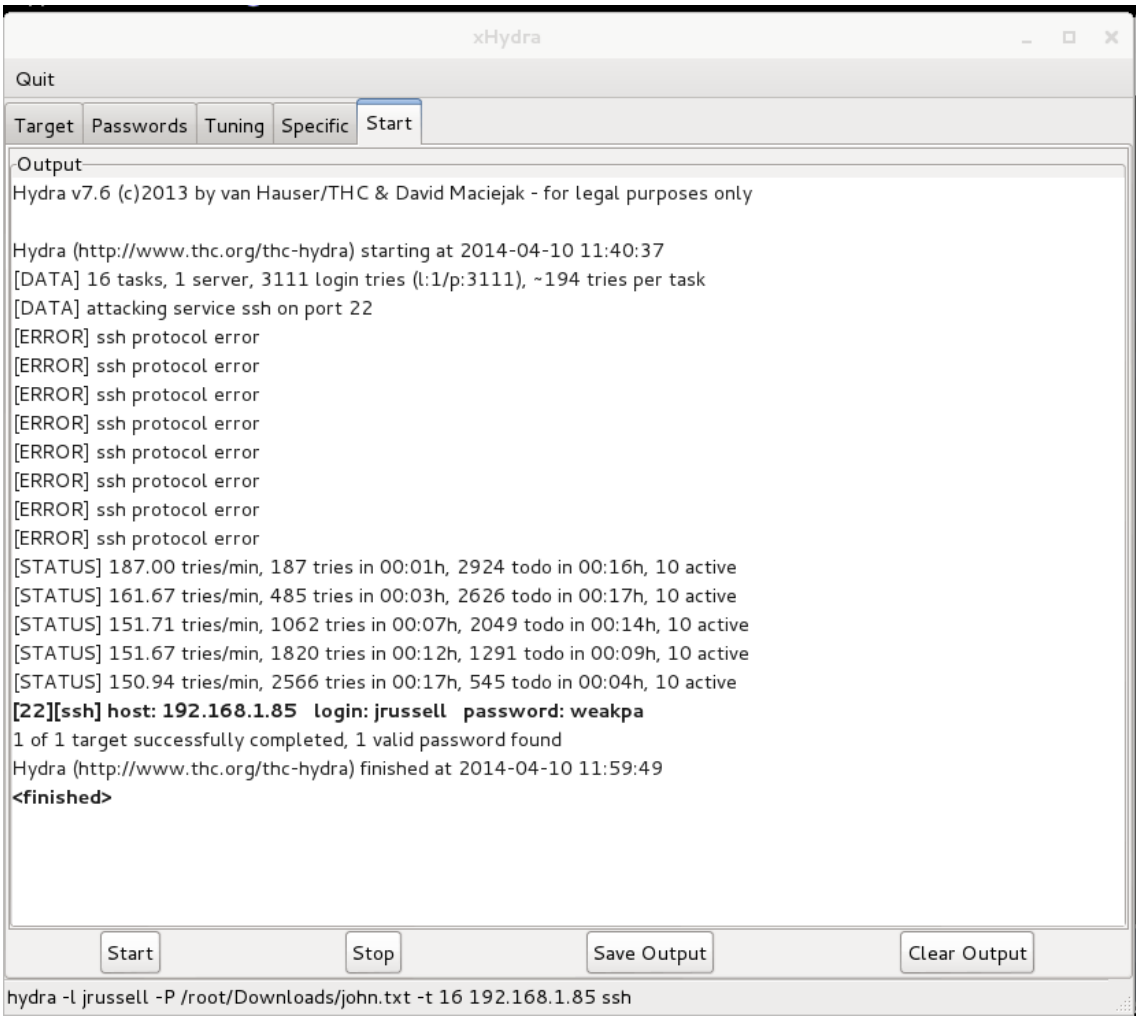

Figure 5. xHydra Dictionary attack speeds

From this, we may also assume that brute force attacks will be limited in the same way as dictionary attacks. They will send similar traffic, only substantially more packets will need to be sent to fascilitate the attack's effective and exaustive process. As such, brute force attacks should be considered only when attacking offline passwords in order to make use of the technique.

The network traffic made by this form of attack is very obvious as shown in Figure 6.

Figure 6. Erroneous SSH traffic

As demonstrated by the traffic in Figure 6, the attack launched is sending very similar sequences of packets over a short time frame. Until a correct password is given, no connection can be made. No key exchange is initiated between the two machines. The system is simply flooded with login requests. On a larger scale than our sample, the difference becomes even clearer as the software sends thousands of these erroneous messages. For comparison, in Figure 7, there is a legitimate SSH login.

```
TCP          66 50976 > ssh [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1811232 TSecr=416192
SSHv2       104 Server Protocol: SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1
TCP          66 50976 > ssh [ACK] Seq=1 Ack=39 Win=29312 Len=0 TSval=1811236 TSecr=416193
SSHv2        98 Client Protocol: SSH-2.0-OpenSSH_6.0p1 Debian-4\r
TCP          66 ssh > 50976 [ACK] Seq=39 Ack=33 Win=5792 Len=0 TSval=416193 TSecr=1811236
SSHv2       850 Server: Key Exchange Init
SSHv2      1338 Client: Key Exchange Init
TCP          66 ssh > 50976 [ACK] Seq=823 Ack=1305 Win=8352 Len=0 TSval=416199 TSecr=1811240
SSHv2        90 Client: Diffie-Hellman GEX Request
TCP          66 ssh > 50976 [ACK] Seq=823 Ack=1329 Win=8352 Len=0 TSval=416199 TSecr=1811248
SSHv2       218 Server: Diffie-Hellman Key Exchange Reply
SSHv2       210 Client: Diffie-Hellman GEX Init
SSHv2       786 Server: Diffie-Hellman GEX Reply
TCP          66 50976 > ssh [ACK] Seq=1473 Ack=1695 Win=33920 Len=0 TSval=1811269 TSecr=416203
```

Figure 7. Genuine SSH login attempt

Attempting to gain access to a user account in this method, using a dictionary, relies on the fact that the correct password is inside the password file. Brute force is a more comprehensive approach. However, based on network speeds, SSH protocol login times, and the obvious nature of network traffic, an online attack is clearly a poor choice in this situation. Instead we will move onto using John the Ripper to attempt an offline attack.

3.1.5 Offline Attack Phase

”The process of cracking Linux and OS X passwords is much the same [Windows] a few slight modifications. Linux systems do not use an SAM file to store the password hashes. Rather the encrypted Linux password hashes are contained in a file called the “shadow” file which is located at /etc/shadow.”[10]

This approach is more indepth, as it requires access to an encrypted password file. The file in question is the shadow file of the Unix operating system. Linux stores user and password data in two files, /etc/passwd and /etc/shadow. Using this method, while any user can request their login information from /etc/passwd, only the root account can access /etc/shadow where the encrypted passwords are stored.

Normally the /etc/shadow file is protected and difficult to access. We used Metasploit Framework in order to exploit the system and it's inbuilt Meterpreter in order to dump the files. Metepreter's thesisation[11] states ”Meterpreter, short for The Meta-Interpreter, is an advanced payload that is included in the

Metasploit Framework. Its purpose is to provide complex and advanced features that would otherwise be tedious to implement purely in assembly." However, Metasploit Framework is a complex topic, far beyond the scope of this thesis. This the methodology of these attacks will not be discussed.

Here is an example of the /etc/password file:

```
cporter:x:1006:1006:Claire,Porter,1,2,3:/home/cporter:/bin/bash

jrussell:x:1007:1007:Jean,Russell,1,2,3:/home/jrussell:/bin/bash
```

And an example of the /etc/shadow file:

```
cporter:$1$8RjArBL6$0beM80wppQxcGdGA9WisQ.:16170:0:99999:7:::

jrussell:$1$QCTLtFy4$hUxTuxL6HrGc0V0hW9qp7.:16170:0:99999:7:::
```

John the Ripper's unshadow command combines the /etc/password and /etc/shadow files to their original form. The form below shows examples of the files before and the combined output file. The command issued is as follows:

```
unshadow ServerPass ServerShadow > ServerUnshadow.txt
```

The results of the command are as follows:

```
cporter:
$1$8RjArBL6$0beM80wppQxcGdGA9WisQ.:1006:1006:Claire,Porter,1,2,3:
/home/cporter:/bin/bash

jrussell:
$1$QCTLtFy4$hUxTuxL6HrGc0V0hW9qp7.:1007:1007:Jean,Russell,1,2,3:
/home/jrussell:/bin/bash
```

This new file, ServerUnshadow.txt was in the correct format to apply John the Ripper's dictionary attack. The command issued:

```
john –word_list=cain.txt ServerUnshadow.txt
```

The attack this time is much faster. Running at over 16000 attempts per second (as demonstrated in Figure 8), a much larger password file (in excess of 300 megabytes) was used and in a far shorter time. Additionally, the GPU can be used instead of the CPU to dramatically increase cracking speed.

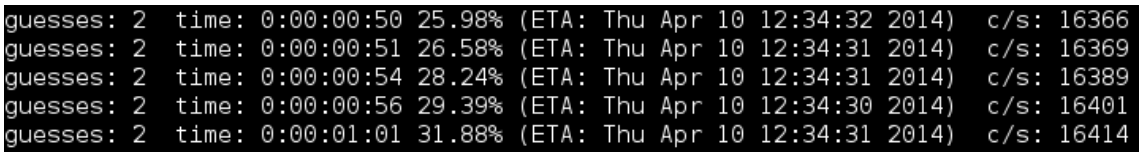

Figure 8. Offline attack speed comparison

It is this result that lends credence to the idea that password-cracking requirements have stayed stagnant while cracking techniques have advanced. Even on a standard processor, a password with 8 characters is surprisingly weak. Sixteen thousand attempts a second would take several years to break into an 8 character password. However, when running on GPU the dramatic increase in speeds renders an 8-letter password mostly useless.

At this point in the experiment, brute force options were considered. However there were several drawbacks that prevented us from running them. Primary among them was the available hardware. While GPUs do drastically increase the speed of cracking passwords, the devices available in our testing environment would not have provided the required power. The cost of obtaining such devices was also unfeasible for testing purposes, as for a reliable result, high end graphics cards would be required.

A second drawback was in the speed of the devices. Running with substandard hardware, if we could attain 200,000 guesses a second, an 8 letter password containing only lowercase letters would still take 12 days to run. While this is not an unreasonable runtime in real world situations, the number of passwords we

needed to test combined with the complexity made testing brute force methods impossible.

For the purposes of this thesis, however, we can consider theoretical values. Brute force attacks can be considered more effective but far slower than dictionary attacks. Both attacks will be subject to similar network traffic when used online and thus our online results only need consideration when it comes to dictionary attacks.

Brute force attacks excel when it comes to offline password cracking on dedicated hardware. As such, we can presume that any system designed for these attacks will provide far better results than our offline dictionary attack.

4 RESULTS AND RECOMMENDATIONS

The results from our testing provide a number of discussion points.

The initial probe of the system revealed a number of vulnerabilities. Primarily SSH was running on the native port of 22. This is considered bad practice purely due to the fact that the service can be easily detected. Obfuscation is an important part of network security. Obscured networks provide an additional level of security beyond technical security measures.

When the SSH service was found to be running on the native port, the decision of how to attack the target became clear. SSH is a common process, in place on almost all servers in production environments, as well as a great number of networking devices. As such, it was a good choice for the attack. However, as SSH connections frequently use certificates in lieu/addition to password security, it is conceivable that a poor password might not be such a large risk, especially in a well-secured environment.

The fact that we were also able to use Meterpreter to dump the shadow file also indicates that this system had security flaws. However, this result is, as was the use of the Metasploit framework, beyond the scope of this thesis.

Logging the network using Wireshark provided a great deal of insight into online attacks. Such attacks are obviously erroneous traffic. Any system administrator with access to network traffic information, for example through intrusion protection systems or network traffic logs, should be able to notice such an attack and blacklist the attacking IP address. Indeed a great deal of software exists to automate this process, such as DenyHosts, the Linux package providing additional security for SSH connections by limiting connection attempts (by default to 4 attempts per minute).

The speed of an online attack also leaves much to be desired. The attacks in our testing phase were launched on 100 megabyte per second connections across a closed network. This is a connection far faster than those to which a standard user would have access. Of course, faster connections exist, but offline techniques are clearly far superior in respect to speed and detectability.

The online dictionary attacks broke passwords after around three thousand attempts, running at less than two hundred tries a minute. Weak or common passwords are far more likely to be contained in a dictionary file. Length and entropy decrease the likelihood of a password being apparent or obvious. These tests rely purely on a password being inside a relative file, however, and these dictionary files can be very large, but the same mathematical logic for a brute force attack applies. A larger password means a larger potential dictionary file and an exponentially increased attack length. The dictionary attack method is clearly better suited to attempting to crack a large number of easy passwords, such as might be dumped from a website.

Offline attacks were far different. They offered better results and an increase in speed even on low-end hardware. Their potential is clear, for attacking stored passwords. The difficulty of this attack is in obtaining the protected password files, usually well-secured on a server, requiring some form of privileged access before the approach can even be attempted.

Brute force attacks are clearly the most reliable method of getting access to a password. However, network speed would clearly be a controlling factor in an online attack. With modern hardware, the computer would be able to guess passwords far faster than the login attempt can be made. Indeed, it could be capable of guessing more passwords than the network can handle. This once again leads to very obvious traffic that should raise alarms with any system administrator.

When looking at the theoretical results outlined in Section 2.2.4, it seems clear that our current password systems are unsatisfactory with regards to the guidelines offered to users. Six to ten letters does not provide enough entropy when faced with modern cracking techniques, even when each possible letter set is used.

Systems which mandate the use of at least two or three of the possible character subsets lead to distinctly more secure passwords than those which do not. Should people insist on using passwords easy to remember, such as a string of words, they should be as long as possible. A word string such as

dogcatrabbitbirdpigsnake is far more difficult to crack than D21Fx0e3 which is an example of a short, high entropy password.

We would suggest the following amended password security guidelines:

- A minimum of 10 letters. 12-15 letters recommended. Each additional letter adds an exponential increase to password strength
- Use all subsets (a-z, A-Z, 1-0, symbols) as each increases entropic value of the passwords
- Passwords should be changed every 90 days in order to mitigate keylogger attacks

REFERENCES

[1] Microsoft Corporation, 2014. Phishing Symptoms [Online]

Location:

http://www.microsoft.com/security/online-privacy/phishing-symptoms.aspx

[Consulted 22/02/14]

[2] Lifehacker, 2013. How to defend yourself against MITM attack [Online]

Location:

http://hackerspace.lifehacker.com/how-to-defend-yourself-against-mitm-or-man-in-the-middl-1461796382

[Consulted 22/02/14]

[3] Symantec Corporation, 2011. The Simplest Security: A Guide to Better Password Practices [Online]

Location:

http://www.symantec.com/connect/articles/simplest-security-guide-better-password-practices

[Consulted 22/02/14]

[4] Facebook, 2015. Security Tips [Online]

Location: https://www.facebook.com/help/379220725465972

[Consulted 23/02/14]

[5] Institute of Electrical and Electronics Engineers, 2012. Measuring Password Strength by Simulating Password-Cracking Algorithms [Online]

Location: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6234434

[Consulted 17/03/15]

[6] Jeremi Gosney @ Passwords^12 Security Conference, 2012. Password Cracking HPC [Online]

Location: http://passwords12.at.ifi.uio.no/Jeremi_Gosney_Password_Cracking_HPC/

[Consulted 23/02/14]

[7] Microsoft Corporation, 2007. A Large-Scale Study of Web Password Habits [Online]

Location: http://research.microsoft.com/pubs/74164/www2007.pdf

[Consulted 26/02/15]

[8] Information Security Forum, 2007. Standards of Good Practise in Information Security. [Online]

Location: https://www.securityforum.org/userfiles/public/SOGP.pdf

[Consulted 15/01/15]

[9] National Institute of Standards and Technology, 2008. SP800-115[Online]

Location: http://csrc.nist.gov/publications/nistpubs/800-115/SP800-115.pdf

[Consulted 24/03/15]

[10] Engebretson, P. 2013. The Basics of Hacking and Penetration Testing.

ISBN 978-0-12-411644-3

[11] Rapid7, 2004. Metasploit's Meterpreter Documentation [Online]

Location: https://dev.metasploit.com/thesiss/meterpreter.pdf

[Consulted 24/03/15]