Journal of Mobile, Embedded and Distributed Systems, vol. V, no. 4, 2013
ISSN 2067 – 4074

WPA/WPA2 Password Security Testing using Graphics Processing Units

Sorin Andrei VISAN
IT&C Security Master
Department of Economic Informatics and Cybernetics
Bucharest University of Economic Studies
sorin_andrei_visan@yahoo.com

***Abstract**:* This thesis focuses on the testing of WPA/WPA 2 password strength. Recently, due to progress in calculation power and technology, new factors must be taken into account when choosing a WPA/WPA2 secure password. A study regarding the security of the current deployed password is reported here.
Harnessing the computational power of a single and old generation GPU (NVIDIA 610M released in December 2011), we have accelerated the process of recovering a password up to 3 times faster than in the case of using only our CPUs power.
We have come to the conclusion that using a modern GPU (mid-end class), the password recovery time could be reduced up to 10 times or even much more faster when using a more elaborate solution such as a GPU cluster service/distributed work between multiple GPUs. This fact should raise an alarm signal to the community as to the way users pick their passwords, as passwords are becoming more and more unsecure as greater calculation power becomes available.

***Key-Words**:* WLAN, IEEE 802.11, WPA/WPA2, PSK, GPU, CUDA, PYRIT

1. Introduction

Wireless Local Area Networks (WLANs) based on the IEEE 802.11 set of standards have been rapidly growing in popularity during the last decade and nowadays it seems almost impossible to have a home user/enterprise environment without an intensely used WLAN. The advantages of these Wireless Local Area Networks such as flexibility, low cost, easy to deploy, continuous improvement and growth of wireless equipment and infrastructure have fueled the embracement of WLAN across the world. However, the widespread of WLANs and the incorrect/poorly implementation of IEE 802.11 have opened a new horizon for digital crime and digital terrorism taking only minutes for an intruder to break in one of these networks if improperly implemented.

2. Problem Formulation

Since wireless communication use an open medium for communication in comparison to wired LANs which by its nature is a vulnerable medium, the 802.11 designers also included shared-key encryption mechanisms: Wired Equivalent Privacy (WEP), Wi-Fi Protected Access (WPA, WPA2), to secure wireless computer networks[1]-[2].
The two security protocols named Wi-Fi Protected Access (WPA) and Wi-Fi Protected Access 2 (WPA2) have been developed by the Wi-Fi Alliance to secure wireless computer networks. After some major flaws and weaknesses were discovered in the previous system, WEP (Wired Equivalent Privacy), The Alliance defined these new standards in response to this threat. In this thesis we will be focusing only on the Wi-Fi Protected Access (WPA, WPA2) mechanisms.
When speaking about security protocols, we need to discuss about the authentication scheme. We will provide below a general and short description of an authentication scheme, covering more details about this field will be done later in this work. In a shared authentication scheme, the authentication is done using a simple challenge and response protocol. First, the station that is trying to authenticate sends an authentication request packet to initiate the communication to which the Access Point will respond with 128 bytes of plain text.

Using the secret shared key, the station will encrypt this plain text and will send the encrypted challenge back to the Access Point. The Access Point then decrypts the received packets and either sends a failure or success frame to the station. This is known as the ”four-way” handshake, the most vulnerable segment of the WPA/WPA2-PSK protocols[1].
WPA/WPA2-PSK is a subset of IEEE 802.11 WPA/WPA2 that skips the complex task of key distribution and client authentication by assigning every participating party the same pre shared key. This master key is derived from a password which the administrating user has to pre-configure e.g. on his laptop and the Access Point. When the laptop creates a connection to the Access Point, a new session key is derived from the master key to encrypt and authenticate following traffic. The "shortcut" of using a single master key instead of per-user keys eases deployment of WPA/WPA2-protected networks for home- and small-office-use at the cost of making the protocol vulnerable to brute-force/dictionary attacks against its key negotiation phase; it allows to ultimately reveal the password that protects the network. In the network discovery/data capturing phase, we have shown how using a network traffic analyzer we can capture vital data in order to begin an attack on our wireless network(a valid four-way handshake is a mandatory requirement).While most of the attacks will work on WPA/WPA2 PSK model(small office/home networks), they will fail on WPA/WPA2 Enterprise(PSK is a low level security for enterprise networks) where further methods of security are implemented (RADIUS server etc.) and one needs to capture the “Enterprise" authentication attempt which is a much more difficult task. When we are referring to a small office or a home network, the 802.11X authentication is not implemented and pre-shared keys are used (PSK). The Primary Master Key (PMK) is obtained by using the algorithm PBKDF2 (Password-Based Key Derivation Function 2). The members participating in the genesis of the PMK are the PSK, SSID(as salt) and its length .These members are hashed 4096 times to generate a 256 bit PMK.
This can be simply illustrated by using the below formula (1):

PMK = PBKDF2(PSK, SSID, SsidLength, 4096, 256)

Later in this document, we will attempt to use CUDA in order to decode a captured hash by brute-force/dictionary attacking all possible combinations using the above cryptographic function until we obtain a matching hash.

```
Reading packets, please wait...

                                Aircrack-ng 1.1

                  [00:02:13] 257820 keys tested (1791.15 k/s)

                           KEY FOUND! [ dictionary ]

      Master Key     : 5D F9 20 B5 48 1E D7 05 38 DD 5F D0 24 23 D7 E2
                       52 22 05 FE EE BB 97 4C AD 08 A5 2B 56 13 ED E2

      Transient Key  : 1B 7B 26 96 03 F0 6C 6C D4 03 AA F6 AC E2 81 FC
                       55 15 9A AF BB 3B 5A A8 69 05 13 73 5C 1C EC E0
                       A2 15 4A E0 99 6F A9 5B 21 1D A1 8E 85 FD 96 49
                       5F B4 97 85 67 33 87 B9 DA 97 97 AA C7 82 8F 52

      EAPOL HMAC     : 6D 45 F3 53 8E AD 8E CA 55 98 C2 60 EE FE 6F 51
root@sorin-K53SD:/home/sorin/Desktop/Captures/test#
```

Figure 1: Password recovery with “Aircrack” (no GPU aid)

ISSN 2067 – 4074

2.1 GPUs and CUDA

The ability to recover a password in acceptable time has been directly linked to the power of an available CPU processing power of a single system. Some application have broken this barrier and made use of distributed coordinated effort using a large number of available computer systems, while others have turned towards hardware acceleration to improve the number of password candidate checks per time. However, these methods have not proven their applicability and efficiency since clusters and hardware accelerators are usually expensive and unavailable for small and medium end users .During the last decade, graphic processing units have become a cost-efficient solution to password recovery. Although these GPUs are used mainly for their display output in the fields of gaming, they could however be used as a high-performance, massively parallel co-processing units, being supported my all-important GPU manufacturers. The GPU aid comes in handy since password recovery is a highly parallelized task.

2.1.1 GPU as Parallel Computers

Since the year 2003, two main trajectories have been established for designing microprocessors. The first one, the multicore trajectory, aims to maintain execution speed of sequential programs while moving into multiple cores. At the beginning, multi-cores were in number of two, but they began to double as each new generation occurred. A current example is the Intel Core i7 microprocessor that has four cores each of the current cores implementing the full x86 instruction set and being capable of hyper-threading with two hardware threads in order to maximize the execution speed of the sequential programs.

On the other hand, the many-core trajectory focuses more on the execution throughput of parallel applications. Like the other trajectory, it first began as a large number of small cores, and once again, it doubled as the generations progressed. A current example is the NVIDIA GeForce GTX 280 GPU with about 240 cores, each core being heavily multithreaded, in order, single-instruction issue processor that shares its control and instruction cache with seven other cores. The main difference between the performance gap between the many-core GPUs and the general-purpose CPUs lies in the different fundamental design philosophies between the two types of processors. The CPU is optimized for sequential code performance making use of sophisticated control logic to allow instructions from a single thread of execution to be executed in parallel or even out of their sequential order while maintaining the appearance of sequential execution. Also, another advantage is the large cache memory that is provided to reduce the instruction and data access latencies. When discussing about performance, memory bandwidth is another aspect to take into account. The GPUs operate at approximately 10 times the bandwidth of the actual available CPUs. On the other hand, the design philosophy of the GPUs is based on the ability to perform a massive number of floating-point calculations. This demand encourages the major GPU vendors to find ways to maximize the chip area and power budget dedicated to floating point calculations. The winning solution was to optimize the execution throughput of massive numbers of threads. The hardware takes advantage of an immense number of execution threads and assigns work even when if some of them are waiting for a long-latency memory access, as a result the control logic required for each execution thread is minimized.

As a conclusion, we can state GPUs are designed as numeric computing engines, thus they will not produce the same performance result when they perform some simple tasks which are designed to perform well on CPUs. As a recommendation, most applications should use both the CPU and GPUs, executing the sequential parts on the CPU and the numerical intensive parts on the GPUs.

2.1.2 GPGPU

General Purpose computing on Graphics Processing Units (GPGPU) is a technique of

using modern graphic processing adapters for non-graphic related computations [3]. A main advantage of using these GPU for solving non-graphic computations would be the fact that GPU consist basically of, compared to CPUs, a large number of arrays of processing units that could be used for massively parallel computations. The whole idea of GPGPU is to distribute parallelized parts of the program's execution from the CPUs to the GPUs in order to obtain a greater speed computations in parallel program parts while CPU resources are made available to other tasks. A condition assumed in GPGPU is data independence. In order for GPGPU to become more user-friendly and embraced by the community of developers, the main GPU vendors have released several Application Programming Interfaces (APIs). One of these APIs is Compute Unified Device Architecture (CUDA) that was developed by NVIDIA in 2006 that offers the developer a friendly and easy to use development framework.

2.1.3 CUDA

As we have discussed above, CUDA (Compute Unified Device Architecture) is a framework developed by the NVIDIA Corporation that comes with an abstraction layer to the actual GPU hardware and allows the programmer to develop GPU assisted applications using the C programming language [3].
When discussing about CUDA programs, we have to take into account the key terms: host, a term that refers to the CPU, and device which refers to the GPU. A CUDA program usually contains mixed C(++) code functions that could be executed on the device or on the host, while functions executed on the device contain only C code and have a special keyword as a function type qualifier. Since there are different memory operation zones referred to as host memory and device memory, a direct exchange of data between the host and devices is not possible during runtime. In order to transfer data from the host to device or vice-versa, we have to issue dedicated memory copy instructions from the host. When writing code for CUDA applications, we usually have to take into account its compiler driver called NVCC. The NVCC task is to separate code that should go to the host's system C compiler and the code that should go to the device. The host's code is straight ANSI C Code and is further compiled with the host's standard C compilers and runs as an ordinary CPU process. The device code is written using ANSI C extended with keywords for labeling data-parallel functions, which are called kernels, and their data structures. This device code is usually furthered compiled by the NVCC and then executed on a GPU device.
These Kernel functions are usually a large number of threads that implement data parallelism. Usually CUDA threads are of much lighter weight than the CPU threads, this is done by the fact that CUDA programmers assume that these threads take very few cycles to generate and schedule due to efficient hardware support while on the opposition CPU threads usually require thousands of clock cycles to generate and schedule.

2.2 Vulnerabilities in WLAN

Wireless networks are vulnerable by definition due to the fact that they use radio frequency which is a broadcast technology as a medium of transmission. The transmission can be intercepted by anyone who is in range. It is thus important to have a secure enough protocol deployed in the wireless network that can assure data confidentiality. Although a secure protocol can be implemented, the management packets are unencrypted and unauthenticated. Without a minimum protection in the management frame, a possible intruder could send the Deauthentication or Disassociation frame to block the request/access to a resource from a valid user, which is known as a Denial of Service (DoS) attack. Even though a Denial of Service attack does not cause such serious damage, it affects the performance critically.

2.2.1 Dictionary Attack

In cryptanalysis and computer security, a dictionary attack is a technique used to try to defeat a cipher or authentication mechanism by trying likely possibilities

ISSN 2067 – 4074

which would reduce the search space of a brute force attack such as words from a dictionary. This method is based on the fact that individuals tend to choose a simple, short and common word from a dictionary. The dictionary attack will try all the combinations of words from a given text file, hash word by word and expect to get the correct PMK.

2.2.2 Rainbow Table

The PMK is calculated from the PSK together with the salt. This gives an interesting aspect to this kind of encryption: assuming that two Access Points should have the same PSK, due to its ciphering mechanism the resulting PMK will be different providing that the two SSIDs of the Access Points are different. A rainbow table is a lookup table containing pre-computed possible keys with the most usually used SSIDs (usually no SSID used, default SSID or the most common names). When we are using the encryption function, the password and salt are hashed 4096 times which is time consuming and results in complication in cracking. In order to save time in cracking, the attacker can obtain a pre-computed file with common passwords and SSIDS, and hash all of them once. This is called a "rainbow table" and a pre-computed 33GB rainbow table is available for downloading on the internet.

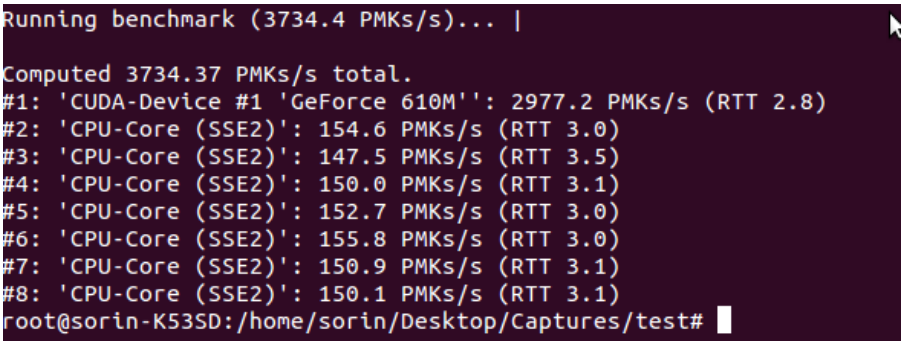

Figure 2. CPU vs GPU PMKs/s benchmark

2.2.3 Denial of Service (DoS)

When speaking of "Denial of Service" attacks, we are referring to an attempt to make a machine or a resource unavailable to its intended users. Referring to our case of study, a denial of service attack can prohibit a legitimated STA to access the networks resources. A "Denial of Service" attack can be performed either by injection of a management packet/ control frame or by using the same frequency ban to jam the network with white noise.

2.2.4 Hardware Acceleration

A Hardware Acceleration attack usually requires a special or modified hardware device that speeds up the cracking process. One of these devices is a Field programmable gate array (FPGA) that is a semiconductor device that can be programmed by source code in hardware or by logic circuit diagram to specify how the chip should word. A single FPGA card can take words 5 times faster than a regular CPU. Another hardware acceleration is the graphic processing unit based password recovery on which we will focus later in this document. We have attached a benchmark that we have obtained on our device in the above picture: our i7 CPU cores computed about 150 PMKs per second, while our old GPU computed almost 3000 PMKs/s.

2.2.5 Deauthentication Attack

It is an attack through which we send disassociation packets to computers/devices connected to a particular WiFi access point. This will disconnect all connected computers from that access point (It won't work if there are no associated wireless client or on fake authentications).This attack is usually used for capturing WPA/WPA2 handshakes by forcing clients to re-authenticate. When a deauthentication attack occurs, the attacker sends a spoofed Deauthentication frame with the forged source MAC address of the Access

Point and the destination the MAC address of the station. The Access Point will not notice the Deauthentication packets since it is being sent directly to the station. A tool that can perform these kind of attacks is ”Aireplay-ng”.

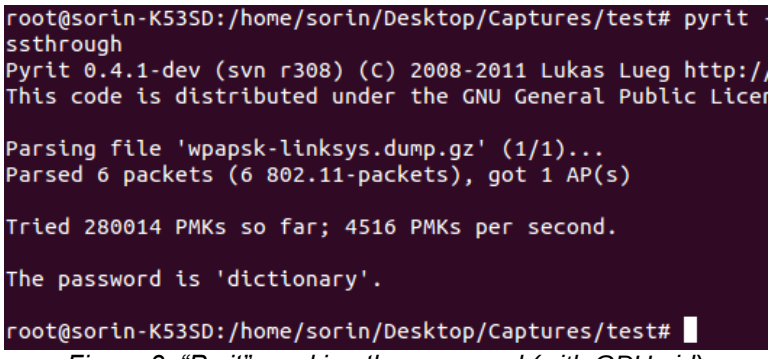

Figure 3. “Pyrit” cracking the password (with GPU aid)

3. Problem Solution

There are three phases that should be deployed in order to penetrate a WPA/WPA2 protected network:

1. Network Data Capturing - the first phase requires intercepting packets from the targeted wireless network in order to get the necessary data for the attack.
2. Parsing Data/Setup Tools - inspect the captured packets to see if they contain a valid four-way handshake. In order to obtain this handshake, a user has to log on to the network. This can be done by forcing disconnection from the network, so that the user has to log on again in order to have access to the network.
3. Key Recovery – after the valid handshake has been captured, using GPUs, we employ the brute-force/dictionary password cracking.

3.1. Network Data Capturing

The first phase focuses on the traffic capture files that are need in order to start our attack against the targeted wireless network. These traffic capture files can be generated using a computer software or hardware that can intercept and log traffic passing over a network. One of these traffic “sniffers” is “Wireshark”, a free and open-source packet analyzer, which is used for network troubleshooting, analysis, software and communications protocol development. In order for our attack to succeed, we must capture a valid “four-way handshake” during the authentication process as we earlier mentioned.

3.2. Parsing Data/Setup Tools

In this phase we will inspect the captured packets to see if they contain a valid four-way handshake. In order to obtain this handshake, a user has to log on to the network. This can be done by forcing a disconnection from the network, so that the user has to log on again in order to have access to the network.

After we have captured our valid four-way handshake, we are ready to set up our CUDA Toolkit that will help us accelerate the process of password cracking. We will assume that we have already installed our NVIDIA drivers according to our CPU architecture. We’ll have to download our CUDA Toolkit, SDK code samples, and developer drivers from the NVIDIA website according to our CPU architecture. After the previous step is complete, we are ready to exploit the functionality of CUDA by using a program named “Pyrit”.

Pyrit allows to create massive databases, pre-computing part of the IEEE 802.11 WPA/WPA2-PSK authentication phase in a space-time-tradeoff. Exploiting the computational power of Many-Core- and other platforms through ATI-Stream, Nvidia CUDA and OpenCL, it is currently by far the most powerful attack against one of the world's most used security-protocols.

ISSN 2067 – 4074

3.3. Key Recovery

The last phase consists of deploying our brute-force/dictionary attack. First, we will use a tool named "Aircrack" to recover the password using only the computational power of the CPUs and then we will use "Pyrit" that uses the GPU power to see what results we will come up with[7].

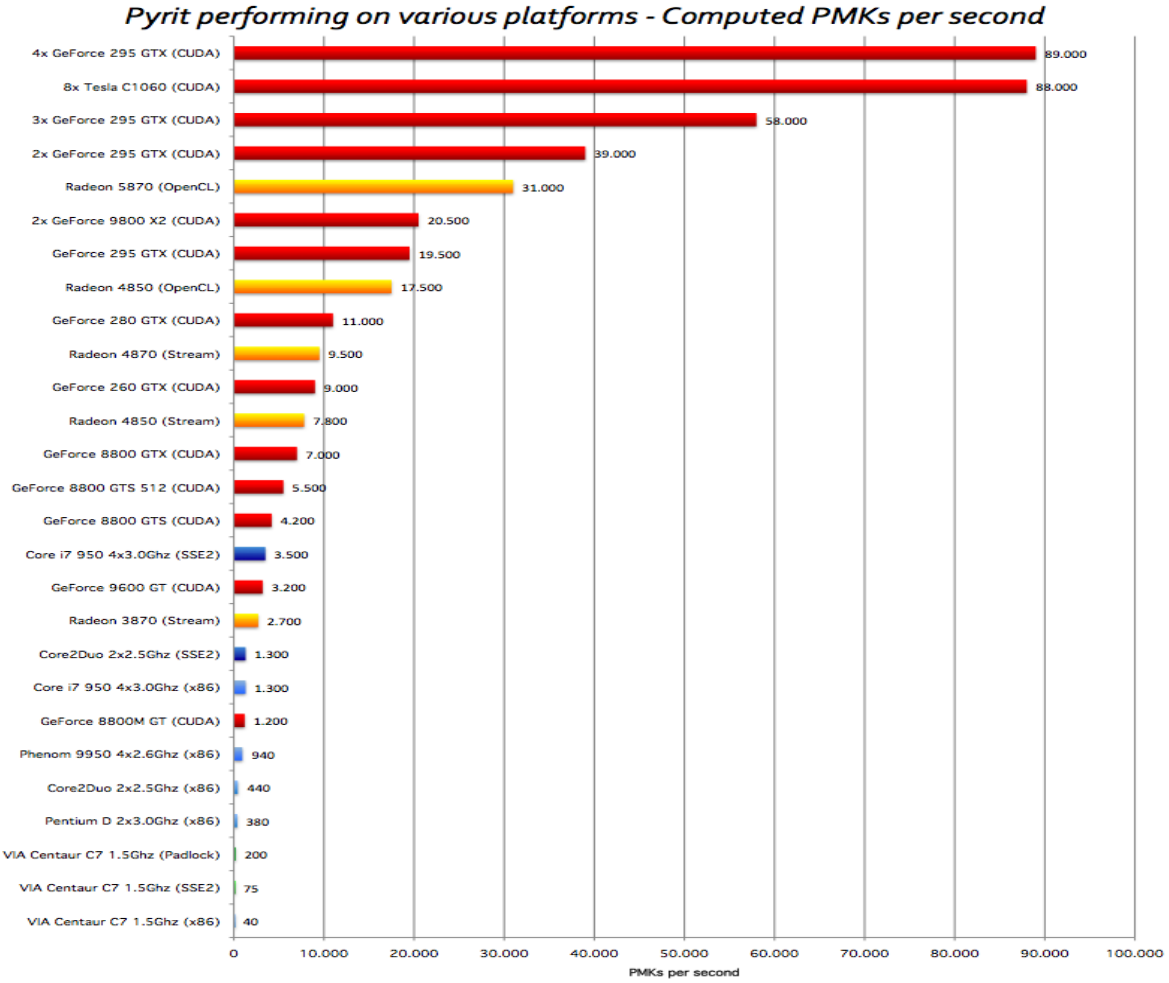

Figure 4. "Pyrit" Performances on different platforms. Source [7]

As we can see, in the first case where no GPU was used, our cracking tool "Aircrack" recovered the password in 2 minutes and 13 seconds from a list of 1.1 million word list with an average speed of 1800 PMKs per second. In the second case, where the GPU aid was used, we can see that the other cracking tool named "Pyrit" recovered our password from the same 1.1 million word list with an average speed of 4500 PMKs per second, almost up to 3 times faster. In our case, we are using an old single GPU, but a cluster GPU service is available online to use in exchange of payment. In an online paper [4], the benchmark used 8 cluster GPU instances and a dictionary of 39 million words. The results are extraordinary. As an average, each GPU core did 47500 PMKs/s with a total sum of about 380.000 PMKs per second, breaking the attacked password in 132 seconds.

4. Conclusion

Having all of the above discussed, we can now proceed in elaborating some conclusions based on the work described in this thesis. The whole procedure of recovering a password can be described in three phases: network discovery/data capturing, setting up all necessary tools

(GPUs toolkit, cracking tools etc.) and the key recovery process itself.
In the network discovery/data capturing phase, we have shown how using a network traffic analyzer we can capture vital data in order to begin an attack on our wireless network (a valid four-way handshake is a mandatory requirement). While most of the attacks will work on WPA/WPA2 PSK model (small office/home networks), they will fail on WPA/WPA2 Enterprise (PSK is a low level security for enterprise networks) where further methods of security are implemented (RADIUS server etc.) and one needs to capture the "Enterprise" authentication attempt which is a much more difficult task.
In the second phase, we have created a tutorial on how to create a CUDA working environment that will aid us in our future work. Also, we have explained how to install some cracking tools.
In our final act, we have used our cracking tools installed in the previous phase in order to recover our wireless password. Using an old generation GPU (NVIDIA 610M graphics card on personal computer), we have shown how a GPU can speed up to 3 times the recovery process (4500 PMKs/s in the case of a single old generation GPU compared to 1500 PMKs/s when only CPU cores were used).The latest GPUs in the present technology can easily achieve 20.000 PMKs/s, not to mention using multiple GPUs or cluster solutions (this will give a 10 times faster cracking speed).
Using an online calculator [6], we have concluded that even with a speed of 20.000PMKs/s, it would take 5 years to break a password of 8 alphanumeric characters one case and 351 years to break a password of 8 alphanumeric characters containing lower/upper case. With a speed of 80.000PMKs/s (assuming the worst case where a GPU cluster service was used), it would still take 14 months to break the password using lower case letters and digits, so a password with the length of 8 letters combined with digits should be sufficiently secured at the moment. (This is the worst case of brute-force attack, there are other effective methods of recovery for example using a dictionary)
One should always take into consideration the following when choosing a password:

1. The minimum length of the password should be about 12 to 14 characters.
2. Passwords should be generated randomly where it is feasible
3. Password should contain lower and upper case characters.
4. Number and symbols should be added to the password if the system permits it.
5. The passwords should not be based on a repeating pattern, common names, number sequences, personal information, and biographical information dictionary words.

As a general conclusion, the fastest way to recover a password is by using GPUs in our aid. At the moment, we can elaborate a general guide-line table in order to create secure passwords, but as technology progresses in time, our current password will become an insecure and obsolete one.

References

[1] Charlene Chow Ying Kwong, "*An Empirical Approach of Wireless Forensics"*, Department of Computer and System Sciences Stockholm University, 2009
[2] N. Tomai, "Particularities of security design for wireless networks in small and medium business (SMB)", *Informatica Economica*, vol. 44, no. 11, pp. 93-98, 2007
[3] Marc Shober, "*Efficient Password and Key recovery using Graphic Cards"*, Ruhr-Universitat Bochum, 2010
[4] Thomas Roth,"*Breaking encryptions using GPU accelerated cloud instances*", in BlackHat Conference, Conf., 2011 [Online] Available: https://media.blackhat.com/bh-dc-11/Roth/BlackHat_DC_2011_Roth_Breaking_encryptions-wp.pdf
[5] David B. Kirk,Wen-mei W. Hwu, *Programming Massively Parallel Processors*, Elsevier Inc, 2010
[6] LastBit Corp, Password Calculator, [Online] Available: http://lastbit.com/pswcalc.asp
[7] Pyrit project, [Online] Available: http://code.google.com/p/pyrit/