JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

PASSWORD CRACKING WITH BRUTE FORCE ALGORITHM AND DICTIONARYATTACK USING PARALLEL PROGRAMING

Mithilesh M Nimbalkar

Guide: Asst. Prof. Prajakta Chowk

Keraleeya Samajam's Model College, Khambalpada Road, Thakurli,

Dombivli (East), Maharashtra

Abstract:

Studying password encryption techniques is important to the information security discipline because it demonstrates the vulnerability of weak passwords and the need for better security measures to protect sensitive information. Although two methods have been develope to crack passwords, both perform different tasks. While brute force algorithms create all the characters together into a common long phr ase, the dictionary has a hard time checking the word list first. This workuses parallel versions of Python, C++ and Hashcat to compare the results of these methods.The results show that NVIDIA GeForce GTX 1050 Ti using CUDA can crack passwords faster than Intel(R).HD Graphics 630 GPU, the password speed is 11.5x and 10.4x respectivelyand . Special characters increase password cracking time and make the cracking process more difficult. Our results showed that theequation greatly increases password processing speed. The br ute force algorithm reaches 1.9x speedup on six cores, while the stop dictionary reaches 4.4x speedup on eight cores with static schedul ing. Research on password encryption techniques demonstrates the need for better security measures to protect sensitive data and the vu lnerability of weak passwords.

Keywords:

parallel computing; password cracking; brute force algorithm; dictionary attack; parallel programming.

Introduction:

In the field of computer security, cross-
linking has proven to be an effective password cracking method using brute force techniques and dictionary attacks. The brute force str ategy involves trying all possible combinations of characters until the correct password is obtained . The computational time required to find a password with brute force cracking techniques depends on many factors, including the length and complexity of the password's c haracter setand the encryption algorithm used to calculate the password number. complexity. Additionally, if the calculation is done on a single processor, it will take more time to crack the password. However, distributing the workload across multiple
processors or devices can help speed up the process. This not only improves password cracking efficiency but also increases the success rate of cracking attempts

which are considered to be more effective, such as a dictionary attack. Dictionary attacks use precompiled lists of

commonly used passwords or word names and compare them to target passwords . This method takes less time than brute force crackin g, but only works if the password is currently in use. Therefore, it is often used in cases where brute force takes a long time to crack a lo ng password . Again, parallel
Appl. Sci. 2023, 13, 5979. https://doi.org/10.3390/app13105979 https://www.mdpi.com/journal/applsci
Appl. Sci. 2023, 13, 5979 22
2 can be used in programming to reduce the time required to execute attacks. However,
connections can also be used to develop
password cracking algorithms, such as using GPUs to process large files. You can wait for someone. believe that the complexity of the algorithm determines how much must be computed. A study by Laatansa et al
. studied the effectiveness of using a GPGPU-based machine to crack SHA-
1 hash passwords using brute force and dictionary attack methods.
The results showed that brute force password cracking was more effective, requiring fewer characters, with 11% of passwords containin g seven characters or less cracked, while translation attacks had a distribution of only 3%. The dictionary attack, on the other hand, pro ved effective in cracking passwords with insecure patterns, cracking 5053 passwords compared to 491 passwords in the best brute force scenario. The combination of two methods (brute force and dictionary) provides a more balanced solution for cracking passwords regar dless of length or security properties . The use of combination codes in password encryption is important not only for security professio nals to strengthen their defenses by detecting weak passwords used by employees, but also for individuals and organizations that may lo se their passwords. Or you forgot your password. Most operating systems
and applications use a key derivation function (KDF) to convert a text password into a hash password to prevent an attacker from obtai ning the cleartext password.
Since KDF is a function, the only way to recover the text password from the hash password will be a brute force attack . It is important to use the same trick in password recovery because it saves time and prevents data and information from forgetting the password. Accor ding to , on average 76% of Internet users use the same password on other websites
. This increases the risk of their account being compromised
. Another password cracking technique is a dictionary attack. A predefined list of passwords is often used in dictionary attacks to ensure a possible
match. This technology poses a threat to many organizations, especially in the area of
accounts and cybersecurity. For example, it may involve using Wi-
Fi networks to gain unauthorized access by targeting traditional passwords . The purpose of this research is to determine the best passw ord cracking method by testing the performance of various hardware configurations against parallel brute force and dictionary attacks.
Additionally, the aim is to investigate the effect of character set
on the performance of passwords, especially those containing special characters.
The results of this study show that hardware configuration
such as more cores or GPU power has a significant impact and can increase the efficiency of effective password protection techniques.
By using these advanced hardware tools, security professionals can increase the speed and accuracy of password cracking, which is ess ential for online attacks. These findings highlight the importance of ongoing research and development in the field of password crackin g hardware configurations and suggest that future advances in this area may have implications for the field of cybersecurity.
The remainder of this study is organized as follows. Section 2 provides a review of 16 relevant articles. The proposed method is present ed in Section 3.
The practical strategy is as follows: Parallel operation of brute force and dictionary attack using Python language, dictionary output aga inst using OpenMP in C++, and brute force and dictionary attacks using Hashcat.
Chapter 4 contains studies on test design and
optimization methods or methods used to control measurement. Section 5 contains results and discussion. A review and comparison wit h other studies are presented in Section 6. Finally, Chapter 7 contains the results and recommendations obtained from the application.

Literature Review:

The use of CUDA computing platform to support brute force algorithms because it requires a lot of money. Applications. education. 2023, 13, 5979 22-bit 3 authors examine five factors that can affect the performance of GPU-based connections . Topics include arithmetic calculations, memory access latencies, data transfers, shared memory, and scratchpad usage. Based on previous research into cracking PDF passwords, they developed custom and testing algorithms to evaluate these features. The final algorithm is created by combining the most influential events. The combined method was implemented on the Tesla C2075 and the results showed a speed of 2.92 for a 2-byte alphanumeric password and 4.77 for a 6-byte alphanumeric password. This study included five factors in both experimental and experimental algorithms, and the results showed that shared memory had the largest impact on algorithm performance. Shared memory helped the algorithm achieve speeds of up to 4.77. In their research, Sarah and Robert learned about the potential of using low-power and energy-efficient devices, electricity, to crack passwords . They believe that such equipment can be used in security applications that do not require immediate operations, such as long-term surveillance or combat. This study was conducted at Adapteva Inc., Lexington, MA, USA, which has the highest performance of 16 GFLOPS/W and low power consumption of 2 W. Describes a proof of concept using the company's Epiphany wafer family. The experiment was carried out using a character set of size 10, resulting in a total of 100 plaintexts of length 4, encrypted using the SHA-512 algorithm. This study compares the performance of a brute force algorithm implemented on a dual-core ARM Cortex-A9 host with the same implementation developed for the Epiphany coprocessor. The results show that using the

same amount of time provides an efficient way to complete the algorithm, providing up to a 16x speedup over the competition. The authors also mention usage limitations such as the minimum amount of RAM available and the need to consider poor memory. Overall, this research provides a proof of concept for energy-efficient password cracking using low-power devices and opens the door to further research in this area. Can et al. proposed an improved brute force password recovery method for SHA512 on GPU using various optimization methods. They use OpenCL to take advantage of the GPU and use C to create programs that run on the GPU. Secure hash (SHA-512) is a one-way hashing algorithm commonly used for password encryption and authentication and is the target of their use in brute force attacks. In their work, they designed a simple GPU system and improved the system using various optimization techniques such as cryptographic defragmentation, register reuse, faster instructions, and matching strategies. The first part covers basic password recovery techniques for SHA-512. In this scheme, the CPU generates the passwords in hardware while the GPU is used to perform the SHA-512 hash calculation. The next step is to make better use of the program, such as reusing the recording and changing some instructions. In the third part, they use algorithms to make the basic process better. After all the tests and research, they managed to reach 1055 M hash value per second on two AMD R9 290 GPUs. Compared to the fastest password cracker Hashcat, its solution is 11% faster. Feng et al. discussed that UNIX systems often use MD5 Crypt encryption technology for authentication. It provides traditional password cracking methods on commonly used computer platforms and ensures system security by using more random salts and greater scheme complexity. However, due to the rapid development of petabyte-scale heterogeneous supercomputer systems such as Tianhe-1, brute force attacks are once again threatening the security of MD5 Crypt. A lot of work has been done on the GPU acceleration platform to increase the speed of MD5 Crypt. However, the memory usage of CUDA architecture has not improved much. Their research investigated this issue and reported a 44.6% performance increase by providing constant memory for writing arrays. In addition, their research using Tianhe-1a, the world's fastest heterogeneous supercomputer, enables efficient implementation of the MD5 Crypt Brute Force attack algorithm. According to the test results, a single operation In , Abdelrahman et al. The use of WPA-WPA2 PSK cracking in network connections is discussed. They run single-
threaded crackers on multiple machines to take advantage of parallel platforms. They also used a translation attack
that depends on the speed of the test to reach the critical point first. The shared strategy is also implemented using Pthreads, OpenMP o n multi-
core CPUs, and CUDA on GPUs. They created a single lockup based on two pieces of data: the password data and the entire four-way handshake CAP data. The password file password
is used to store the password and the CAP file
is used to provide the login information needed to crack it. Using the CUDA
C language extension, they applied a broken device to the GPU, allowing users to take advantage of
CPU and GPU processing power. The results show that the use of multi-
core processors and GPUs in systems with integrated platforms increases performance by 16x and 41x, respectively. Again Ref. propos ed a new method to crack Wi-
Fi passwords using GPU and parallel processing. Currently WPA/WPA2 protocol is more secure but still vulnerable to brute force. This algorithm
effectively improves the efficiency of Wi-Fi password cracking by limiting the combination of translation data
and using GPU and CPU to split water activities simultaneously.
Use dictionary crunching speed between CPU and GPU together
Much faster than traditional CPU computing power. However, this method is ineffective for
long passwords.
David et al. [16] tries to speed up the execution of the hash function and compare strings, which can take a long time. However, these o perations can be simplified since each password can be checked independently. High performance computing (HPC) is available for bet ter performance. GPU
computing can increase performance. More GPUs can be used for further scaling, but this increases communication latency and therefo re reduces overall performance. In this study, MPI was used to reduce communication delays and manage data communication between systems. This article introduces three password recovery algorithms using both MPI and CUDA. These algorithms differ in terms of GP U memory usage and data allocation. Algorithms involving classification of dictionaries and password files have been shown to be very efficient, achieving 17x and 12x speedups respectively using four by eight GPUs.
minimum memory algorithm exhibits slow performance due to communication delay. These algorithms scale well across multiple GPU s and can be used with larger libraries. This program can increase computer security by identifying weak
passwords and can be used to process large files using MPI and CUDA.
Anh-
Duy et al. proposed an isomorphic parallel brute force algorithm using GPU to crack passwords. They use CUDA framework 3.0 to cra ck SHA1.
They also divide passwords into five groups. The first group contains only numeric characters, the second group contains lowercase cha racters, the third group contains numbers and lowercase letters, the fourth group contains both lowercase and uppercase letters (signific ant characters of the alphabet), and the last group is numbers and characters. sensitive. . They measured the password cracking time for each group.
The password consists of six characters hashed using SHA1. Additionally, the proposed algorithm
also has the disadvantage of using memory. Testing was done on a 240-core Tesla C1060 with a 1.3 GHz core clock and a 480-
core Tesla C2050 with a 1.15 GHz core clock. Using a Tesla C2050, they cracked the code to six digits in less than 1 second.
Maruthi et al. [18] argued that although the FPGA implementation of the MD5 hashing algorithm is faster than its software equivalent, a preimage brute force attack of the MD5 hashing algorithm theoretically requires 2128 iterations. Their research focuses on acceleratin

g brute force attacks against the MD5 algorithm. For generation use. education. 2023, 13, 5979 22
5/5 MD5 hash uses all 64-layer pipelines to generate predicted
passwords based on our architectures. MD5 hash generator and password generator
pairs are parallelized using 32/34/26 samples to find hashed passwords using MD5 techniques. Total performance of approximately 60 t rials per second is achieved using a single Virtex-7 FPGA chip.
In , Laatansa et al. It is believed that most password files are hashed. Due to various human errors and weaknesses
equipment may fall into the hands of unauthorized persons and even malicious persons. Brute force and dictionary attacks are examples of possible attacks against cryptographic devices using GPGPU-
based systems. The researchers' work describes the effectiveness of brute force and dictionary attacks against SHA1 hash passwords usi ng GPGPUbased machines. Research results show that for shorter passwords (more than 11% consist of 7 characters or less), brute forc e methods are more successful at cracking passwords than a dictionary. password. The dictionary attack is more effective at cracking pa sswords with insecure patterns (5053 passwords) compared to the best brute force attack scenario requiring 491 passwords. No matter h ow long or secure the password, greater password compliance methods can be achieved using a combination of techniques (brute force + dictionary).
In , Qingbing and Hao published an article on recovery password for WinRAR3
encrypted archives without filename encryption. Current cracking methods use either a CPU or GPU platform and are limited by slow d ecision and decompression algorithms on the CPU and GPU, respectively.

Table 1. summary table of different criteria

Ref.	Dataset	Number of Samples	Number of Threads/Processors	Techniques	Best Result
[12]	None.	100 plain texts of length 4.	Parallel implementation of a brute force algorithm for 16 processors/threads. the Epiphany co-processor.		Speedup of up to 16 times that of theserial version.
[6]	None.	None.	They used several optimization techniques: combination of passwords, 2560.	repetition of register utilization, faster instructions execution, and meet-in-middle.	1055 M hash/s.
[13]	None.	None.	Each node of Tianhe-1A supercomputer has two CPUs, one GPU. The CPU has 6 cores while the GPU has 448 cores operating.	Using both the CPUs and the GPU on one single node.	326,000 MD5 hashes are searched per second, which is 5.6 times faster than the performance of the CPU-only version.
[14]	None.	2.	The computer has an Intel i7-4710HQ processor (2.50 GHz, 4 physical cores, 8 threads). The multi-core version was tested on a different computer (Intel i7-2630 QM, 2 GHz, 4 cores, 8 threads) using Ubuntu 12.04. The GPU used was a GeForce GTX 860 M.	They used the shared memory model.	They archive the best score by using GPU platform on the Windows operating system with 1,000,000 passwords in 384 s.
[15]	None.	N/A.	The server has two 16-core CPUs, 192 GB memory, and four GeForce GTX 1080 graphics chips, each with 8 GB video memory.	Hybrid parallel processing using Multi-CPU-GPU for calculations.	The performance of a single-core GPU is about 80 times higher than that of a single-core CPU.

Password settings are created and protected using MD5 encryption. We use the same password as the previous method. The sequence a

	'assword String	Iash Value (MD5)
'assword 1	bc14	80eaaf275cf1d34b88d0b8c6c7da20b
'assword 2	d180	ecce0ac22fc85a9899a1f8ba2c08bfb
'assword 3	7ro1	fd4d72214cd938a1bef4e1a58f4366f
'assword 4	ynq0	ea15dcb8862ccab2fa6388fb43317f6
'assword 5	3kli	70244cdc900710194338673e26dba1f
'assword 6	1m5e	c4a21024aa58f2558a1e98e5839e54d
'assword 7	1z5l	9db818049350277c4400cb01dd3f112
'assword 8	sdf32	b697d6d9fbd75cb15fb4670c5aaf0ca
'assword 9	5d3a1	7c818c75041578020d71acd4c2ea79f
'assword 10	3g3v1	be9b447aa3d00aefcc69629b626d460

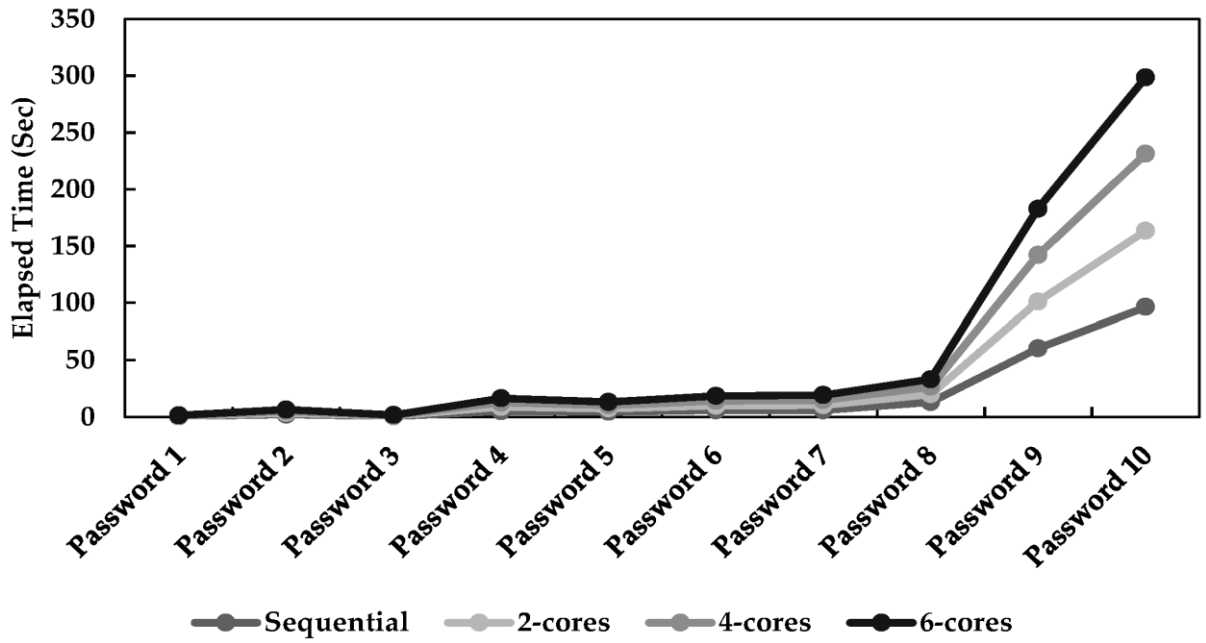
nd corresponding hash value of each password are shown in Table 3. We will try to crack the password by feeding only the hash value t o the Hashcat tool. Hashcat is used by specifying a script or command line with different parameters depending on the test objectives. T he "m" parameter specifies the encryption type, in this case 0 (MD5). The "a" parameter determines the type of attack to be performed, 3 indicates a brute force attack and 0 indicates a dictionary attack. The "d" parameter selects the device used for password cracking, "3" represents CPU + GPU 0, and "1" represents GPU 1 (CUDA) specific. Finally, determine the character of the test, "l" parameter is the English digit, "u" is the English character, "d" is the digit and "s" is the designed special character. Our goal is to test the impact of addi ng special characters on password performance. We also aim to evaluate the effectiveness of various hardware configurations to perfor m these attacks. Table 4 shows the scripts used in different cracking scenarios.

Results and Discussion:

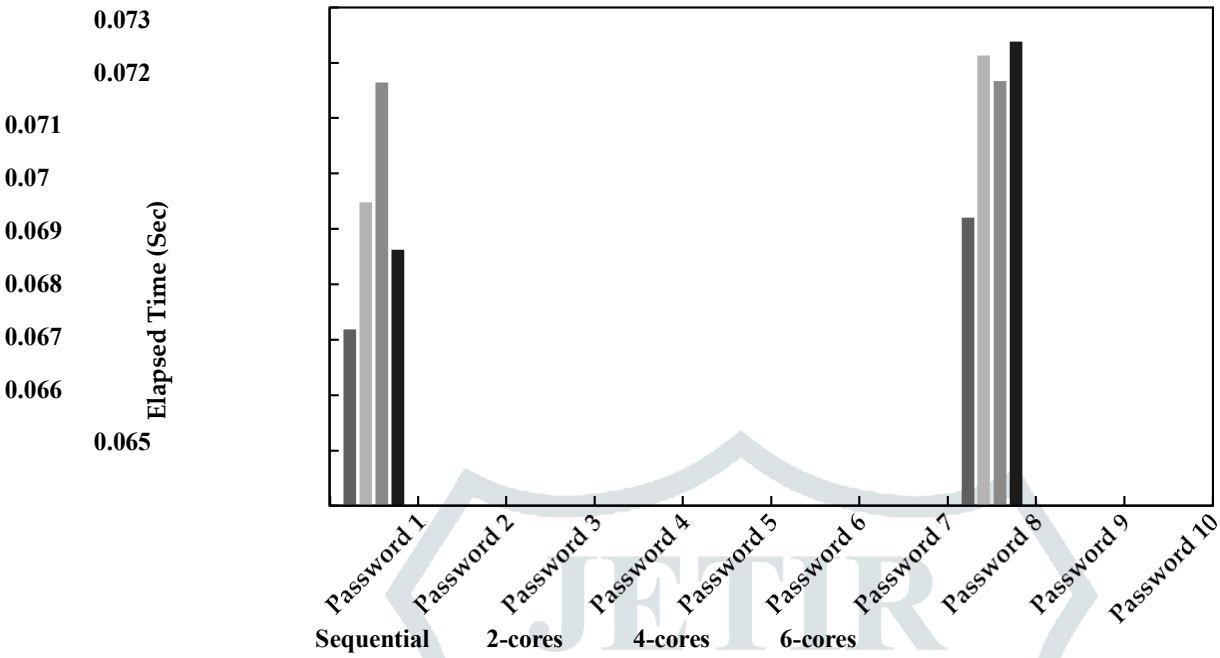
The first method and the second method were tested on a computer with the following specifications: Intel(R) Core (TM) i59400F CPU , 16 GB RAM and NVIDIA GeForce RTX 2060 Graphics card. The code in both technologies is written in Python. Randomly generate a list of ten passwords as a string containing lowercase English characters and numbers, with a maximum length of six characters. Pass words at the end of the list are more difficult than the passwords at the beginning. What makes password cracking even more difficult is if they contain numbers (passwords with numbers at the beginning of the string are particularly difficult) or English characters. The org anization or order of characters in a character set has a significant impact on password cracking ability.Advanced character systems, suc h as character sets that prefix characters in the English alphabet followed by numbers, make it difficult to tell one apart, using brute forc e techniques to predict the future. The characters of the character initially determine the password. Using the brute force method of shor t passwords, we achieve better scores on adjacent numbers than on even numbers. For example, the serial code of the first password (th e simplest password) took 0.2471 seconds, and the corresponding number took 0.3286 seconds. But the longer and more complex the p assword, the more effective the matching code will be compared to the matching code. The best improvement is achieved when using p arallel numbers with the highest hardware count (for cores). The speed ratio is 1.9 times. Figure 11 below shows the implementation of the first met

Ten various passwords being tested for the Brute Force Attack.

On the other hand, when using the second method, “Dictionary Attack”, we decided to compare them by testing the same list of passwo rds used together with the brute force method. As we mentioned before, the total number of passwords in the dictionary is close to 14 m illion. Since the tested passwords are generated (not frequently used), only two passwords can be cracked in the dictionary against the s ystem. No improvement was found between parallel and linear code in Python as the processing is done by the "ProcessPoolExecutor" l ibrary. This process, which involves breaking the work into pieces (editing the source code) and distributing it across multiple cores, tak es some time and resources and can cause additional work or overhead, which ultimately hinders the good results of doing the same thi ng. However, the execution time of the dictionary attack is approximately 930 times faster than that of the brute force attack. Figure 12 below shows the results of the dictionary attack test.Actually, dictionary attacks are tintensive versus brute force attacks due to the smal l number of iterations available (about 14 million iterations in the database here). However, a brute force attack can capture all target pa sswords. While the length of the password has a positive effect on the execution time in a brute force attack, it is possible to easily crac k a long password in the dictionary. The first 6-
core password dictionary attack technique is approximately 4.79 times faster than the brute force attack. Cracking eight passwords is ab



out 930 times faster because it uses brute force.



Ten various passwords being tested for the Dictionary Attack.

In the third set, the tests were run

on a computer with the following features: 11th Generation Intel(R) Core(TM) i51135G7, 8 GB RAM and Intel(R) Iris(R) Car Graphic s. Experimental results show that the worst performance for parallel processing in terms of execution time occurs on the eightcore static scheduler. Processing time increased by 4.4 times from 181 seconds to 41 seconds. However, the fewer the number of blocks used in d ynamic scheduling, the slower the execution time compared to sequential code. In other words, the more blocks there are in a dynamic t ime, the higher our speedup over a sequential program. The chart below shows the different settings and time taken for ten different pas swords. Figure 13 below shows the results of the third operation.We also make sure that the output includes the iteration number where the password is located (equal to the password index in the dictionary) and the phone number where the password is located. The time u sed to enter the password and more importantly to run the thread. The screenshot below captures one of the tests for dynamic time eval uation of four blocks with eight threads. For these four blocks, the program divides the password list into four equal parts, each execute d by a separate thread. This means that objects are processed equally across all four threads; This can result in faster execution times. H owever, if a thread completes some of its work before other threads, it does not need to wait until the thread completes its work because it is dynamic, not normal, where it should be. waiting for the rest of the execution Thread completed. Complete their part.In our fourth approach, we aim to examine the password cracking efficiency of different types of GPUs using the wellknown Hashcat tool, as shown in Figure 14. Specifically, our results show that GPU 0 (Intel(R) HD Graphics 630) took 23 seconds to crack the specified list of passw ords at 254.8 MH/s according to Hashcat parameters, while GPU 1 (NVIDIA GeForce GTX 1050 Ti) did the same task only He comple ted it in 23 seconds. 2 seconds, speed 2558.3 MH/s. This means GPU 1 can crack mobile password
up to 11.5x faster even if the CPU supports GPU 0. Regarding the character set containing special characters, the combination of CPU a nd GPU GPU 0 took 197 seconds to crack the password list, while GPU 1 needed only 19 seconds to perform the same task. Therefore, on the task of cracking a list of passwords, GPU 1 is approximately 10.4 times faster than the combi nation of CPU and GPU 0. Our results show that the time required to crack the password increas es significantly when special characters are added to the Hashcat buffer. Specifically, the combination of CPU and GPU 0 took 197 seconds to complete the task; this was longer than the 23 seconds required to crack the same list name password without special characters. This shows that including special characters in password s makes the cracking process even more difficult. Similarly, it took GPU1 19 seconds to enter the password when

special characters were included, compared to 2 seconds when nopecial characters were included. Table 5 below shows the brute force attack test results.

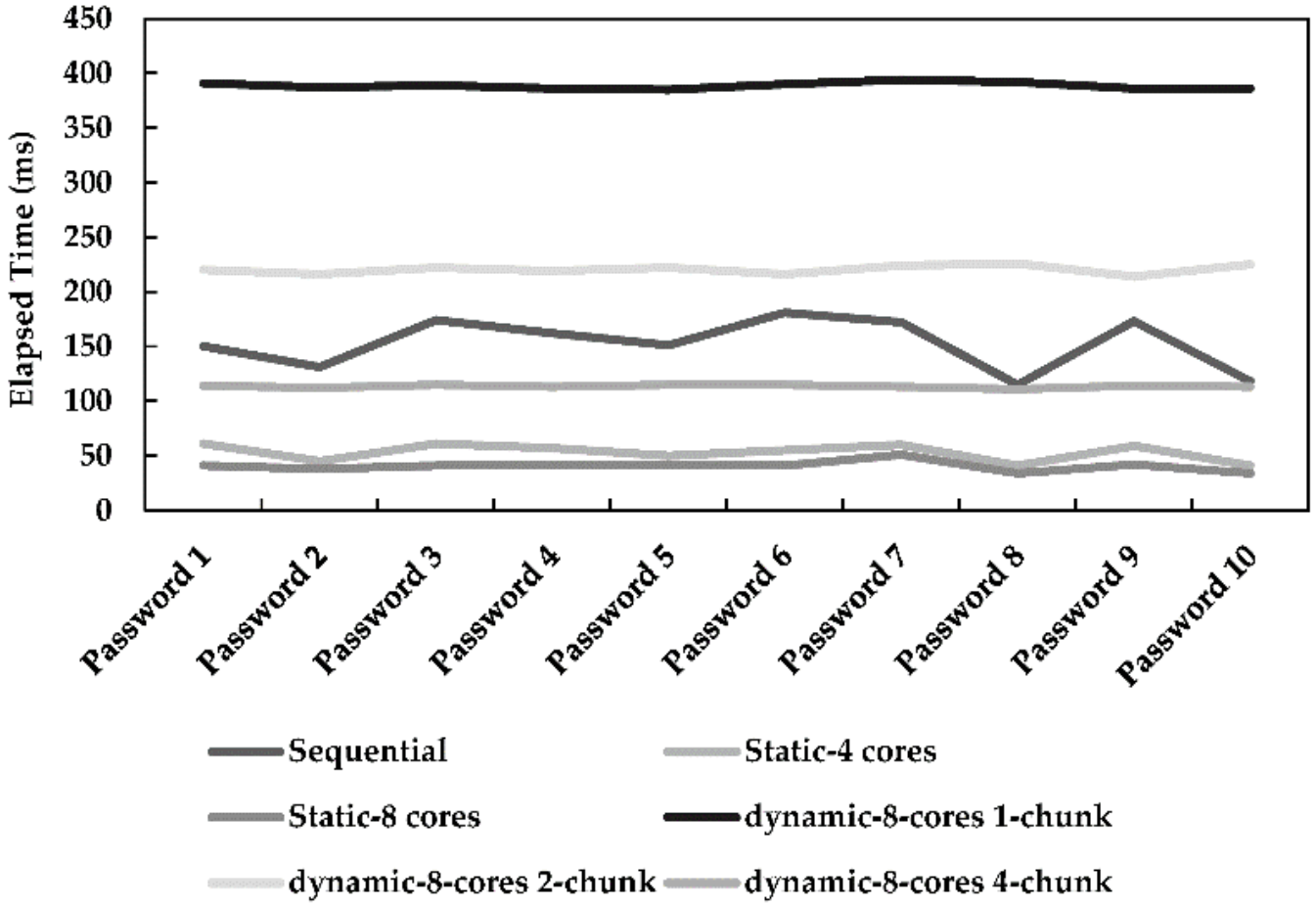
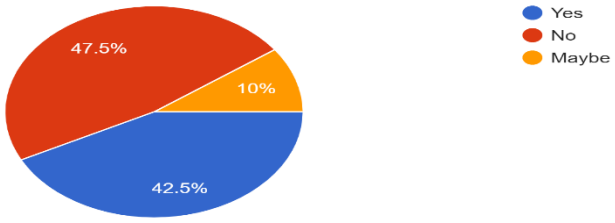
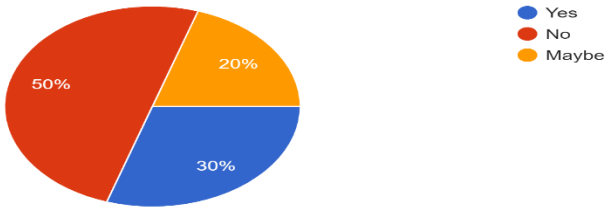


Chart and survey results:

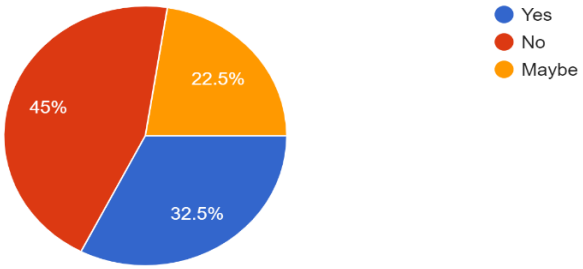
Are you aware about password cracking ?
40 responses



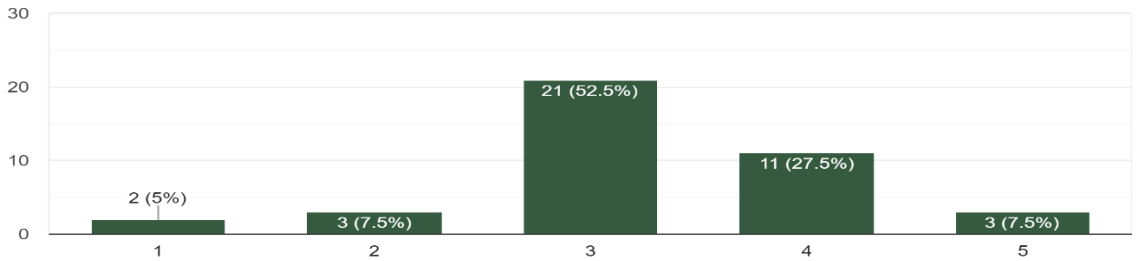
Does ethical hackers good for our society ?
40 responses



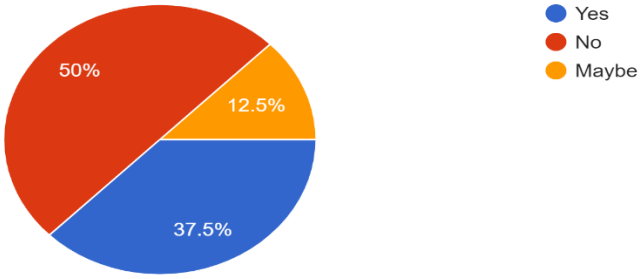
Have you ever heard the term "Brut force attack" ?
40 responses



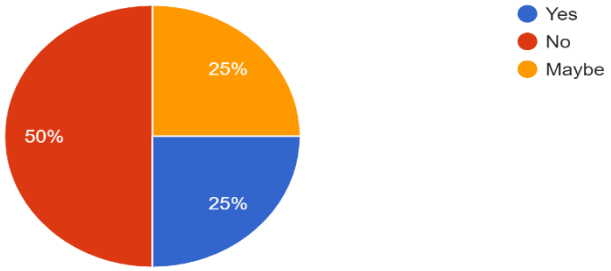
How much do you aware about hacking ?
40 responses



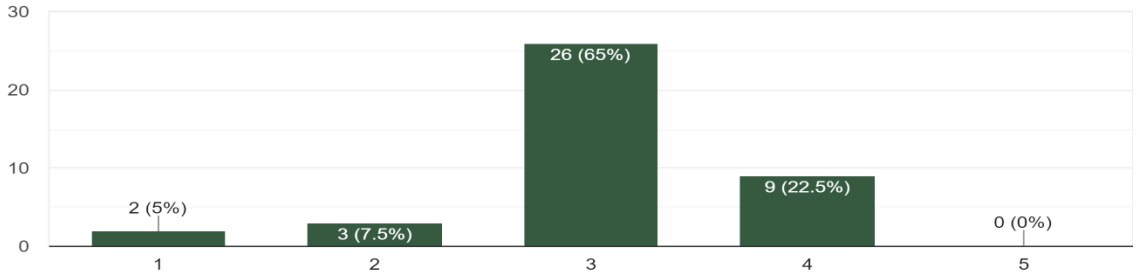
Will you like to know how password cracking works ?
40 responses



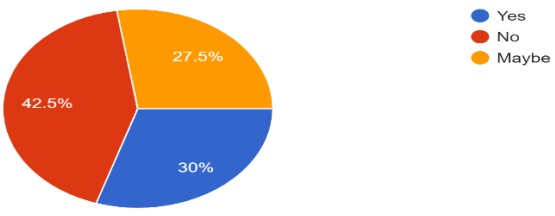
Do you think password cracking is a good skill ?
40 responses

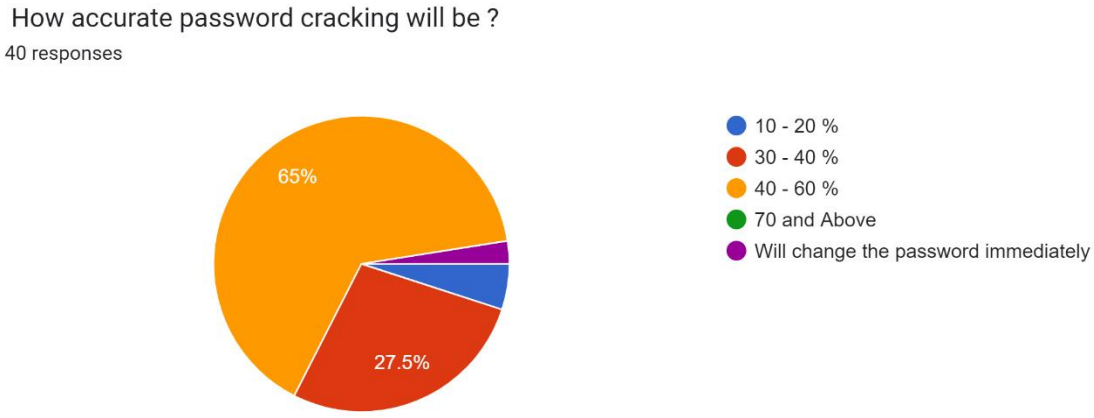


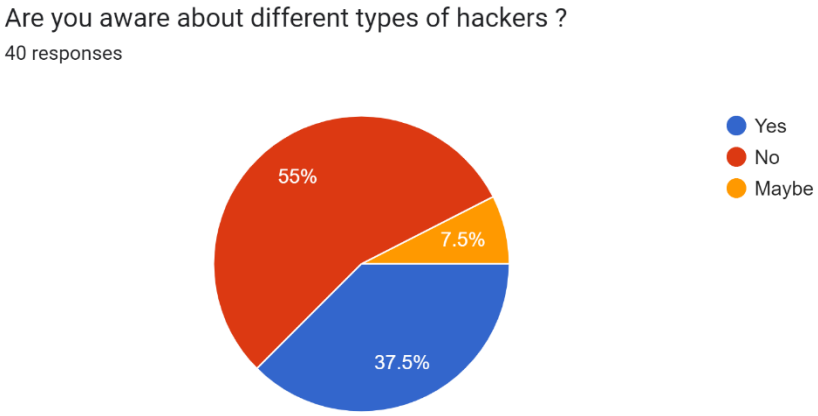
How much do you feel safe while logging into various sites ?
40 responses



Do you think hackers are dangerous to our community ?
40 responses







References :

1. Grover, V.; Gagandeep. An Efficient Brute Force Attack Handling Techniques for Server Virtualization. *SSRN Electron. J.* **2020**. [CrossRef]
2. Liu, P.; Li, S.; Ding, Q. An Energy-Efficient Accelerator Based on Hybrid CPU-FPGA Devices for Password Recovery. *IEEE Trans.Comput.* **2018**, *68*, 170–181. [CrossRef]
3. Tirado, E.; Turpin, B.; Beltz, C.; Roshon, P.; Judge, R.; Gagneja, K. A New Distributed Brute-Force Password Cracking Technique. In Proceedings of the Future Network Systems and Security: 4th International Conference, FNSS 2018, Paris, France, 9–11 July2018; Springer: Cham, Switzerland, 2018; pp. 117–127.
4. Hranický, R. Digital Forensics: The Acceleration of Password Cracking. Ph.D. Thesis, Brno University of Technology, Brno,
Czechia, 2022.

5. Swathi, K. Brute Force Attack on Real World Passwords. *Int. J. Res. Public Rev.* **2022**, *3*, 552–558.
6. Ge, C.; Xu, L.; Qiu, W.; Huang, Z.; Guo, J.; Liu, G.; Gong, Z. Optimized Password Recovery for SHA-512 on GPUs. In Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Guangzhou, China, 21–24 July 2017; IEEE: Piscataway, NJ, USA, 2017; Volume 2, pp. 226–229.
7. Laatansa; Saputra, R.; Noranita, B. Analysis of GPGPU-Based Brute-Force and Dictionary Attack on SHA-1 Password Hash. In Proceedings of the 2019 3rd International Conference on Informatics and Computational Sciences (ICICoS), Semarang, Indonesia, 29–30 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–4.
8. Zhang, Z.; Liu, P. A Hybrid-CPU-FPGA-Based Solution to the Recovery of Sha256crypt-Hashed Passwords. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2020**, *2020*, 1–23. [CrossRef]
9. Hranický, R.; Matoušek, P.; Ryšavý, O.; Veselý, V. Experimental Evaluation of Password Recovery in Encrypted Documents. In Proceedings of the ICISSP, Rome, Italy, 19–21 February 2016; SciTePress-Science and Technology Publications: Setúbal, Portugal, 2016; Volume 2016, pp. 299–306.

10. Nakhila, O.; Attiah, A.; Jinz, Y.; Zou, C. Parallel Active Dictionary Attack on WPA2-PSK Wi-Fi Networks. In Proceedings of the Proceedings—IEEE Military Communications Conference MILCOM, Tampa, FL, USA, 26–28 October 2015; IEEE: Piscataway, NJ,USA, 2015; Volume 2015-Decem, pp. 665–670.
11. Hendarto, I.L.S.; Kurniawan, Y. Performance Factors of a CUDA GPU Parallel Program: A Case Study on a PDF Password

Cracking Brute-Force Algorithm. In Proceedings of the 2017 International Conference on Computer, Control, Informatics and its Applications (IC3INA), Jakarta, Indonesia, 23–26 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 35–40.
12. Mount, S.; Newman, R. Energy-Efficient Brute Force Password Cracking. In Proceedings of the 2015 European Intelligence and Security Informatics Conference, Manchester, UK, 7–9 September 2015; IEEE: Piscataway, NJ, USA, 2015; p. 189.
13. Wang, F.; Yang, C.; Wu, Q.; Shi, Z. Constant Memory Optimizations in MD5 Crypt Cracking Algorithm on GPU-Accelerated Supercomputer Using CUDA. In Proceedings of the 2012 7th International Conference on Computer Science & Education (ICCSE), Melbourne, Australia, 14–17 July 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 638–642.
14. Abdelrahman, A.; Khaled, H.; Shaaban, E.; Elkilani, W.S. WPA-WPA2 Psk Cracking Implementation on Parallel Platforms. In Proceedings of the 2018 13th International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt, 18–19 December 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 448–453.
15. Qabalin, M.K.; Arida, Z.A.; Saraereh, O.A.; Wu, F.; Khan, I.; Uthansakul, P.; Alsafasfeh, M. An Improved Dictionary Cracking Scheme Based on Multiple GPUs for Wi-Fi Network. *Comput. Mater. Contin.* **2021**, *66*, 2957–2972. [CrossRef]
16. Apostal, D.; Foerster, K.; Chatterjee, A.; Desell, T. Password Recovery Using MPI and CUDA. In Proceedings of the 2012 19th International Conference on High Performance Computing, Pune, India, 18–21 December 2012; IEEE: Piscataway, NJ, USA, 2012;pp. 1–9.
17. Vu, A.-D.; Han, J.-I.; Nguyen, H.-A.; Kim, Y.-M.; Im, E.-J. A Homogeneous Parallel Brute Force Cracking Algorithm on the GPU. In Proceedings of the ICTC 2011, Seoul, Republic of Korea, 28–30 September 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 561–564.
18. Gillela, M.; Prenosil, V.; Ginjala, V.R. Parallelization of Brute-Force Attack on MD5 Hash Algorithm on FPGA. In Proceedings of the 2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID), Delhi, India, 5–9 January 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 88–93.
19. Ji, Q.; Yin, H. Speedup and Password Recovery for Encrypted WinRAR3 without Encrypting Filename on GPUs. *J. Phys. Conf. Ser.*
2020, *1673*, 12047. [CrossRef]

20. Ding, Q.; Zhang, Z.; Li, S.; Liu, P. Energy-Efficient RAR3 Password Recovery with Dual-Granularity Data Path Strategy. In Proceedings of the 2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 26–29 May 2019;IEEE: Piscataway, NJ, USA, 2019; pp. 1–5.

21. Hu, G.; Ma, J.; Huang, B. Password Recovery for RAR Files Using CUDA. In Proceedings of the 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, Chengdu, China, 12–14 December 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 486–490.
22. Pi, J.; De, P.; Mueller, K. Using Gpus to Crack Android Pattern-Based Passwords. In Proceedings of the 2013 International Conference on Parallel and Distributed Systems, Seoul, Republic of Korea, 15–18 December 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 450–451.
23. An, X.; Jia, H.; Zhang, Y. Optimized Password Recovery for Encrypted RAR on GPUs. In Proceedings of the 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems, New York, NY, USA, 24–26 August 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 591–598.
24. Niu, H.; Wu, B.; Wang, Q.; Zhu, Z. Research on Steel Barrel Flattened Seam Recognition Based on Machine Vision. *J. Phys. Conf. Ser.* **2020**, *1633*, 12014. [CrossRef]
25. Digman, E.S.; Orantoy, R.S.; Velasco, J.A.; Blanco, M.C.; Regala, R.; Cortez, D.M. Enhancement of Hakak's Split-Based Searching Algorithm through Multiprocessing. *Int. J. Innov. Sci. Res. Technol.* **2022**, *7*, 1068–1072.
26. Norouzi, M.; Wolf, F.; Jannesari, A. Automatic Construct Selection and Variable Classification in OpenMP. In Proceedings of the ACM International Conference on Supercomputing, Phoenix, AZ, USA, 26–28 June 2019; pp. 330–341.
27. Burns, W.J. Common Password List (Rockyou.Txt). Available online: https://www.kaggle.com/datasets/wjburns/common-password-list-rockyoutxt (accessed on 15 February 2023).
28. Alnoon, H.; Al Awadi, S. Executing Parallelized Dictionary Attacks on Cpus and Gpus. *Moais. Imago Fr.* **2009**. Available online: https://moais.imag.fr/membres/jean-louis.roch/perso_html/transfert/2009-06-19-IntensiveProjects-M1-SCCI-Reports/HassanShayma.pdf (accessed on 7 March 2023).