

# A Novel Approach for Distributed Password Cracking Based on Command Line

Kaushal Shah<sup>a</sup> and Dhaval Vasava

*School of Technology, Pandit Deendayal Energy University, Gandhinagar, India*

**Keywords:** Password, Hash Cracking, Distributed Cracking, Dictionary Attack, Brute Force Attack, Online Attack


**Abstract:** Password cracking process still continues to remain prevalent during a pentest or a digital forensic investigation. In both cases, time is critically important. Most of the password cracking tools available today can be used to crack different types of hashes and encryption media. Though these tools utilize system components to enhance their performance, still it might not crack it in the required duration of time. This process of cracking passwords could be accelerated with the help of distributed password cracking technique. This paper aims at studying and analysing some of the popular password cracking tools and to propose a new command-line distributed password cracking tool which aims to resolve some of the issues with current distributed password cracking tools.

## 1 INTRODUCTION

Password based authentication has been there for a very long time and though the security provided by such an authentication mechanism solely depends on the user: the character set used by user while creating the password, the password length, the uniqueness of the password, the guess ability of the password through social engineering; it's still widely used and it unlikely to be replaced by other authentication mechanism in upcoming few years (Hranicky', Zabal, et al. , 2020) (Pervan, Knezovi', et al. , 2022), (Pervan, Knezovi', et al. , 2019), (HRANICKY' , 2022), (Jourdan and Stavrou, 2019). Therefore, it is of utmost importance to set a strong password every time during registration.

Passwords which are created without using any password policy are generally considered to be weak because a user will most likely set a password which would be easy for him/her to remember and so the user ends up either having a password of small length or would select a well-known password and in both cases, the attacker would crack those type of passwords in few seconds with the help of some of the popular password cracking tools which are freely available today. Therefore, nowadays it has become very common to use a password policy while

registering a user for a service and in that way users will be forced to create a strong Password for themselves (Siponen, Puhakainen, et al. , 2020), (Guo, Zhang, et al. , 2019). The most basic password policy requires the password to be of minimum length 8, the character set used for creating password should contain at least one lowercase alphabet, at least one uppercase alphabet, at least one digit and at least one special character. As today's users have a large presence in the online world which ranges from social media platforms to bank accounts, creating and remembering each and every password is extremely difficult for a user and so users may end up using the same password for different accounts. Though this problem can be easily resolved using a password manager, there is a risk of losing everything once the password manager itself is compromised. And also with password manager, the passwords created are so random in nature that remembering even a few of them can be nearly impossible for some users. Like every coin has two sides, password cracking too can be used in a good way and in a bad way and it solely depends on the user's intention. While an attacker can use a password cracking tool for gaining access to someone's account/system, a penetration tester can use the same tool for auditing the password policy of a company and its users. These password cracking tools can prove to be very helpful during password

<sup>a</sup> <https://orcid.org/0000-0002-8494-9752>

recovery of an account/system either hit by malware or even in simple cases like when a user forgets his/her password or during forensics. There are many different password cracking techniques like brute forcing, dictionary based attacks, hybrid attack which involves a combination of both brute forcing and dictionary based attack, rainbow cracking method for hash cracking and even through the use of social engineering techniques (Rudy, Rodwald, et al. , 2020), (Zion, 2018), (Shah, Patel, et al. , 2020), (Shah and Jinwala, 2019) Some of the existing popular password cracking tools incorporate these techniques to provide flexibility to the user while cracking a password. Some of these tools utilize system components like CPU and GPU for effective password cracking. Even though with such powerful tools sometimes it takes a very long time to brute-force a password. Though one can improve system configurations like allocating more powerful CPUs, GPUs, RAM, storage etc but it then increases the cost drastically. Some major pen-testing organizations have a dedicated rig for password cracking but other organizations, researchers or pen-testers who would like to test a system against a password cracking tool can't simply afford such a rig. Having a distributed password cracking tool in such cases will aid in testing the system on the same level as that of the rig but with minimum cost compared to that of the rig. Utilizing processing power of other hosts on the network will not only speed up the process but it also means that every host will have a lower load compared to when a single host utilizing all its resources for the task of cracking a password which will result in decrease of system performance during other task done by user while the password cracking task is running in background.

### 1.1 Our Contributions

We have reviewed some of the existing tools carried out in the area of password cracking, which mostly revolves around popular tools like Hashcat, John The Ripper and THC Hydra, and distributed password cracking, which involves tools like Fcrackzip and Hashtopolis. Based on the analysis of these tools, we proposed a new light-weight command-line distributed password cracking tool which covered some of the flaws of the current password cracking tools.

## 2 LITERATURE REVIEW

A review of work done in the area of password cracking was carried out. It involves going through some of the existing password cracking tools like hashcat, JohnTheRipper and Hydra to learn about their working, some of the novel techniques they use to crack passwords and other features which make them top choices for anyone in the cyber world in the area of password cracking. Some of the existing distributed password crackers like Hashtopolis and Fitcrack have also been considered as part of the literature review. Yisa et al. (Yu and Yin, 2021) reviewed some of the top open-source password cracking tools. The review paper starts with discussing different forms of password authentication and then the different types of password cracking techniques like dictionary attacks, brute-force attack, hybrid attack and rainbow tables. Later, for every password cracking tool that was taken into consideration, a brief overview and features were also listed out. Some of the best practices for password protection were also mentioned which includes the use of salting for storing passwords, use of strong passwords which are formed by combination of capital letters, small letters, numbers and special characters and the use of multi-factor authentication. Pahuja and Sidana (Pahuja and Sidana, 2021) compared different password cracking tools from an implementation point of view. The authors considered three tools for their study. Besides considering pre-installed tools like Hashcat and John The Ripper, they have also considered another tool called Fcrackzip, which is different from other two, as this one only focuses on zip password cracking. John The Ripper and Hashcat mainly focuses on hash cracking but the former could be used to crack zip files with the help of additional tools like zip2john. A comparative analysis of above mentioned tools was also provided in a tabular form, which showed the functionality/working of the tools as well as the complexity associated with each of those tools.

Kakarla et al. (Kakarla, Mairaj, et al. , 2018) have discussed a few password cracking tools with an emphasis on THC Hydra. They started with first discussing different password cracking techniques then proceeded to explain the technicalities of THC Hydra by demonstrating attacks on FTP, SSH and SMTP servers. Lastly, few protective measures that are taken against password cracking were also discussed. Hranicky et al. (Hranicky, Zabal, et al. , 2019), in their paper, have first discussed the need for distributed password cracking. They used the self-proclaimed fastest cracking tool, Hashcat, to

develop a distributed Hashcat tool using the BOINC framework called Fitcrack. In their literature survey, they considered many works that were done in the area of distributed cracking but only considered Hash- topolis for comparing their work as at that time Hashtopolis was the only one, which was a popular, well-maintained and open-source solution for distributed computing, as mentioned in their paper. Then, the architecture of their proposed work was discussed with a brief discussion on each of their components. The tool incorporated different distribution strategies for different types of attack modes. Their analytical results and comparison, shows that Fitcrack performs better than Hashtopolis.

Password cracking tools like Hashcat (Jens Steube (Steube, 2015)), John The Ripper (openwall (openwall, 2011)) and THC Hydra (vanhauser the (Yisa, Baba, et al. , 2016), each have their own set of features that make them popular even today. Hashcat supports over 300 types of hashes and it is the world's fastest password cracker, as mentioned in their official website. John The Ripper can easily identify the hash type of the hash to crack, whereas for hashcat we need to have the hash type of the hash and then we can proceed to cracking that hash. Though, there are other separate tools like hash-identifier (Zion3R ) which can help identify the hash but that feature is currently not available by hashcat. John The Ripper can also crack zip files and ssh keys with the help of other separate tools zip2john (openwall, 2011c) and ssh2john (openwall, 2011). THC Hydra supports many different protocols that one can use to conduct an online password cracking attack. There isn't a single tool which incorporates all of these capabilities and therefore our work somewhat focuses on doing so. Though existing distributed password tools like Fitcrack and Hashtopolis can significantly improve the password cracking time, their installation and setup process is quite lengthy and could be somewhat complex for newbies. Also, as a node is entirely dedicated to the server, we need to have at least two nodes to work with it. We can definitely use Hashcat for a single node, but that means that we need to have more than one tool in our toolbox. In this paper, we try to eliminate this issue, by developing a command-line lightweight distributed password cracking tool in python3, which could work both as a standalone application as well as a distributed password cracking application.

### 3 PROPOSED WORK

A light-weight tool written in python3 which could serve as both a standalone application (like hashcat, hydra etc) as well as a distributed application (like Hashtopolis, Fitcrack, etc) for the purpose of password cracking. By default, the tool will work as a standalone application but it can be used as a distributed application by simply providing the appropriate flag/argument. The node/host could either be used as a server machine or as a client machine when using the tool as a distributed application. Currently, it incorporates the following attack techniques to crack the password : dictionary based at- tack, bruteforce based attack and online dictionary attack. The distributed password cracking process starts with the hash type identification phase as shown in Fig 1, in which the hash type of the provided hash is categorized into the following hash types : MD4, MD5 and the SHA-1, SHA-2, SHA-3 families, based on the hash length of the provided hash(es). A character set check ensures that the entered input is indeed a hexadecimal string and a valid hash. The type of identification can result in having multiple hash types for a provided hash. For example, a hash of 512 bits, could either be a SHA512 hash or SHA3512 hash. In such cases, the tool takes both of the hash types into consideration and so for

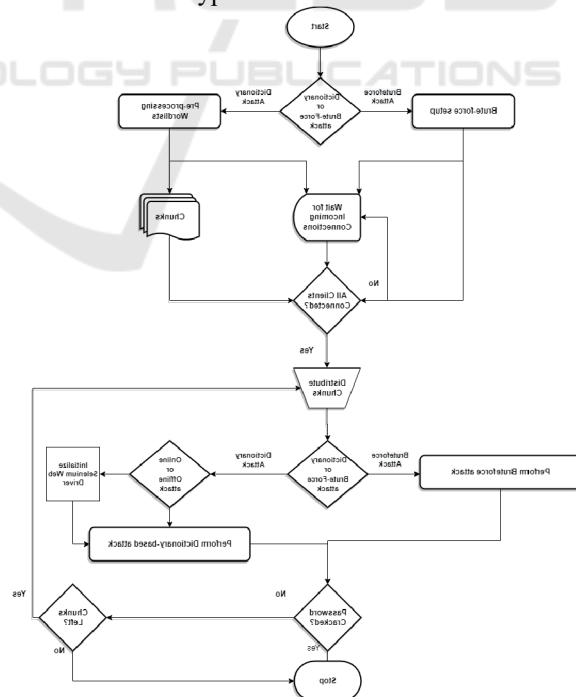


Figure 1: Flowchart of the proposed scheme

each guessed password, it calculates hash digest for both the hash types and compares both of them with the provided hash. After this hash type identification phase, depending on the attack technique, wordlist pre-processing phase in case of dictionary attack or brute force setup phase in case of brute-force attack starts. The entire process of the proposed password cracking work is depicted in Figure 1.

### 3.1 Dictionary Attack

The process of wordlist-based attack starts with the phase of pre-processing of wordlist(s). Support has been provided to process and use multiple wordlists in order to crack the hash. The preprocessing phase includes reading the wordlist(s) and merging them to generate a single wordlist (with no duplications of words). That wordlist is then divided into smaller chunks. The reason for having a pre-processing wordlist phase is because, some of the popular wordlists like 'rockyou' and '10-million-password-top-list-1000000.txt' by 'seclist', contain millions of words and it is very likely that they have a high percentage of overlapping words and this could result in overall increase in the cracking time of the hash. Also, there is no point in repeating the words in a password cracking process. The wordlist pre-processing process usually starts in a different thread and leaves the main thread for the server to listen for incoming connections from the clients. This way, the overall performance of the program is improved especially when there are a number of wordlists and a number of clients to connect.

After the wordlists pre-processing phase and after all the clients are connected, the server sends the chunk of data to clients, which includes the hashes that need to be cracked, the chunk that needs to be used and the attack technique that needs to be used for cracking the hash and some other configurations. Each client including the server itself will participate in the cracking phase and will utilize the chunk they received to crack the hash. If the hash is cracked by any of the clients, then it will send a message to the server informing it about the same, along with the cracked password. The server will then update its list of hashes left to crack and will send that updated hashes list to others in the network, when they send a message re- requesting the next chunk of data. This process will continue until there are no hashes left to crack or there are no chunks left with the server. At the end of this whole process, the server will close all the connections and list all the passwords cracked during the complete process.

### 3.2 Brute-force Attack

This process starts with the brute-force setup phase on the server side, in which, based on the given configuration options/arguments such as pass- word pattern/charset, minimum and maximum password length, all possible characters at the first/start/leftmost position of the word is guessed and then a list of tuples is generated, where the first item of the tuple would be one of all the possible characters that were guessed for the first position, whereas the second argument/element in the tuple will indicate the length of strings that are to be generated starting with the character at the first item. For example, (a,8) indicates that the node having this tuple will have to generate all the strings starting with character 'a' and of length '8'. After this setup phase, the cracking phase begins where each node in the network will take one tuple from the list of tuples and will perform the brute force attack. As the process of handling client and bruteforcing are on separate threads (Python-Software-Foundation, 2012b), the server will also participate in the cracking phase. Unlike dictionary attack, which is a I/O bound task, brute-force attack is a CPU bound task and so we have used the concept of multiprocessing (Python-Software-Foundation, 2012a) for increasing the performance of the system. By default, it uses 10 processes to do the work. As with the dictionary attack, this process will continue until there are no hashes left to crack or there are no tuples left with the server. In case a hash has been cracked, the client and the server will work the same way as it was in the dictionary attack. At the end, all the connections are closed and all cracked passwords are listed by the server.

### 3.3 Online Attack

For the online attack mode, the flow will be similar to that of a dictionary- based attack except that instead of cracking the hash, this time it will be working on cracking the password of the login page of a web app. For such an attack, it uses selenium under the hood. Selenium not only takes care of the cookies, but it also takes care of the other security elements like the hidden fields of a web app, if any. And this was the reason selenium was preferred ahead of the 'request' module, which is generally used when dealing with requests in Python. The user only needs to provide the target url, the username and the wordlist to work with. The server will then send the chunks that were generated during the pre-processing wordlist phase to clients along with the target url and the username.



Each of the nodes, including server, will then initialize the cracking process by running the browser through selenium in headless mode and it will then find the appropriate location in the web page (target) for entering the username and the guessed password from the wordlist. As selenium requires appropriate drivers to run a browser, another module 'webdriver-manager' was used to download the necessary drivers for selenium to work with. The process in case of any successful login, will be the same as it was for the successful hash crack process in brute force and dictionary attack except that the server will not inform others on the network as it did in other two attacks. The process ends when there are no chunks left to work with.

## 4 RESULTS AND ANALYSIS

For comparative analysis of the tool, two distinct physical machines were used, one acting as a server and the other as the client machine. As shown in the table 1, two completely different nodes with different architecture and OS were used to carry out the test. Both nodes also differ in memory space allotted to them to see if there is any performance issue because of it. The nodes were communicating with each other on a home network, rather than having a completely isolated environment as in most of the real-world scenarios we would have regular traffic going through the network. And so it's important to test the performance of these tools in such an environment.

Table 1: System Configuration of server and client machine

	Server Machine	Client Machine
OS	Ubuntu 20.04.5 LTS x86 64	Kali GNU/Linux Rolling x86 64
Kernel	5.15.0-52-generic	5.16.0-kali7-amd64
CPU	AMD Ryzen 5 4600H with Radeon G	Intel i5-2300 @ 2.793GHz
GPU	VMware SVGA II adapter	VMware SVGA II adapter
Memory	3924 MiB	2938 MiB

For dictionary attack, some of the current popular wordlists were used, like 'rockyou.txt' (which is already available in Kali Linux under the '/usr/share/wordlists/' directory) and all the wordlists by seclists (Miessler, 2012) under the passwords category. The results of this test are shown in table 2. The cracking time of Hashtopolis and DPC (proposed tool) were noted on using different wordlists. And the

hash of the last word in the list was used in each of the experiments. Note that the time noted was in the format : 'hh:mm:ss'. The category 'all' in table 1 indicates that all the wordlists from SecLists (it includes the following wordlists : '10-million-password-list-top-100.txt', '10- million-password-list-top-1000.txt', '10-million-password-list-top-10000.txt', '10-million-password-list-top-100000.txt', '10-million-password-list-top-1000000.txt', '10-million-password-list-top-500.txt', '100k-most-used-passwords-NCSC.txt', '10k-most-common.txt', '1900-2020.txt', '500-worst-passwords.txt', 'SplashData- 2014.txt', 'SplashData-2015-1.txt', 'SplashData-2015-2.txt', 'best1050.txt', 'best110.txt', 'best15.txt', 'common-passwords-win.txt', 'medical-devices.txt', 'top-20-common-SSH-passwords.txt', 'top-passwords-shortlist.txt', 'worst-passwords- 2017-top100-slashdata.txt') + 'rockyou.txt' were used.

As seen in table 2, in some cases the proposed DPC tool works better than Hashtopolis, while in some cases the Hashtopolis is better than the DPC tool. Thus, we can say that they are very close to each other in terms of performance.

Table 2: Dictionary based attack comparison

Wordlist	No of words	Hash algorithm	Hashtopolis	DPC
10-million-password-list-top-1000000.txt	999,998	SHA1	09s	5s
rockyou.txt	14,344,392	MD5	29s	32s
10-million-password- list-top-1000000.txt + rockyou.txt	15,344,390	SHA1	8s	6s
10-million-password- list-top-1000000.txt + rockyou.txt	15,344,390	MD5	31s	40s
All	15,613,886	SHA1	3m 45s	33s
All	15,613,886	MD5	3m 10s	37s

In cases where multiple wordlists were used (as in 'All' category - where a total of 22 wordlists were used), DPC performs way better than Hashtopolis because of its wordlist preprocessing phase (where repeated words among wordlists are eliminated), as seen in Figures 2 and 3.

Comparison with FitCrack was not possible, because the errors during installation and setup of

FitCrack client were not resolved within the duration of this experiment. Thus, we can say that between Hashtopolis and DPC, the latter performs better when it comes to working with multiple wordlists.

SHA1

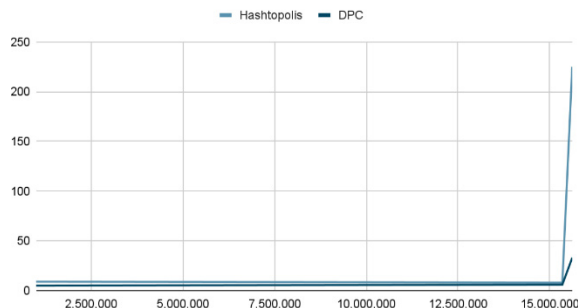


Figure 2: SHA1 hash cracking performance comparison

MD5

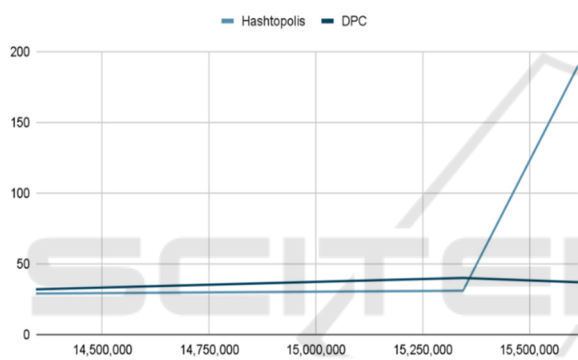


Figure 3: MD5 hash cracking performance comparison

## 5 CONCLUSIONS

Even today passwords remain one of the most dominant authentication methods. And so security of passwords still plays a vital role in overall security of the organization. Mainly, weak passwords are the ones that are cracked easily while strong passwords take relatively longer time to crack. The time to crack may vary depending on factors like password length, character set used to create password, the hash type used to encrypt, the hardware configuration of the system used for cracking etc. Password cracking process, if successful, can become a valuable factor in deciding the outcome of a cyber investigation. And to further speed up the process of cracking, we can make use of distributed password cracking tools. Tools like Hashtopolis and FitCrack which are built on top of hashcat do pretty good work when it comes to distributed password cracking, provided they are already installed and configured properly to use. But when it comes to instantaneous use, these tools take a

considerable amount of time for installation and initial setup. As the proposed tool is written in Python and is bound to be used as a command line tool, it is simply a matter of time to install and it's ready to use. The server also participating in cracking helps in cases where there are not more than two nodes available. And also gives the tool the ability to work as a standalone application. Merging wordlist features of this tool can prove to be useful when working with multiple wordlists at a time. And with support for online attack (http/https protocol), it stands out among the other tools.

For making a one-stop solution for password cracking, an extended work to make it feature-rich application can be done. Various features like adding support for other hash types, support for other protocols and support for GPU based cracking can be added to enhance the capabilities of the tool. In order to make sure that the server and client communicate in a secure fashion, efforts can be made in using ssh protocol for communication.

In cases where the security/investigation team have weak infrastructure, distributed password cracking tools can prove to be very helpful, as it uses the computing power of multiple nodes for cracking which in turn depicts the behaviour of a high-end system. Hence, we can say that there is a need for distributed password cracking tools considering the challenges we face today in penetration testing and digital forensics.

## REFERENCES

- Dave, D., Kava, M., Gupta, R. K., and Shah, K. (2021). Deep learning approaches for intrusion detection system. In 2021 IEEE International Conference on Technology, Research, and Innovation for Betterment of Society (TRIBES), pages 1–6. IEEE.
- Guo, Y., Zhang, Z., and Guo, Y. (2019). Optiwords: A new password policy for creating memorable and strong passwords. *Computers & Security*, 85:423–435.
- HRANICKY', R. (2022). Digital forensics: The acceleration of password cracking.
- Hranicky', R., Zabal, L., Ry'savy', O., and Kol'a'r, D. (2019). Distributed password cracking with boinc and hashcat. *Digital Investigation*, 30:161–172.
- Hranicky', R., Zabal, L., Ry'savy', O., Kol'a'r, D., and Miku's, D. (2020). Distributed pcfg password cracking. In *European Symposium on Research in Computer Security*, pages 701–719. Springer.
- Jens Steube, G. G. (2015). Hashcat - advanced password recovery. <https://hashcat.net/hashcat/>. (last accessed on 20/11/22).
- Jourdan, P. and Stavrou, E. (2019). Towards designing advanced password cracking toolkits: Optimizing the

- password cracking process. In Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization, pages 203–208.
- Kakarla, T., Mairaj, A., and Javaid, A. Y. (2018). A real-world password cracking demonstration using open source tools for instructional use. In 2018 IEEE International Conference on Electro/Information Technology (EIT), pages 0387–0391. IEEE.
- Miessler, D. (2012). Common-credentials. <https://github.com/danielmiessler/SecLists/tree/master/Passwords/Common-Credentials>. (last accessed on 20/12/22).
- openwall (2011a). Johntheripper. <https://github.com/openwall/john>. (last accessed on 22/11/22).
- openwall (2011b). ssh2john. <https://github.com/openwall/john/blob/bleeding-jumbo/run/ssh2john.py>. (last accessed on 05/12/22).
- openwall (2011c). zip2john. <https://github.com/openwall/john/blob/bleeding-jumbo/src/zip2john.c>. (last accessed on 03/12/22).
- Pahuja, D. and Sidana, P. (2021). Implementing and comparing different password cracking tools. International Research Journal of Engineering and Technology.
- Pervan, B., Knezović, J., and Guberović, E. (2022). Energy-efficient distributed password hash computation on heterogeneous embedded system. *Automatika*, 63(3):399–417.
- Pervan, B., Knezovic, J., and Pericin, K. (2019). Distributed password hash computation on commodity heterogeneous programmable platforms. In 13th USENIX Workshop on Offensive Technologies (WOOT 19).
- Python-Software-Foundation (2012a). Multiprocessing. <https://docs.python.org/3/library/multiprocessing.html>. (last accessed on 10/12/22).
- Python-Software-Foundation (2012b). Threading. <https://docs.python.org/3/library/threading.html>. (last accessed on 10/12/22).
- Rudy, J. and Rodwald, P. (2020). Job scheduling with machine speeds for password cracking using hashtopolis. In International Conference on Dependability and Complex Systems, pages 523–533. Springer.
- Shah, K. and Patel, D. (2020). Exploring the access control policies of web-based social network. In ICDSMLA 2019: Proceedings of the 1st International Conference on Data Science, Machine Learning and Applications, pages 1614–1622. Springer.
- Shah, K. A. and Jinwala, D. C. (2019). Novel approach of key predistribution for grid based sensor networks. *Wireless Personal Communications*, 108:939–955.
- Siponen, M., Puhakainen, P., and Vance, A. (2020). Can individuals' neutralization techniques be overcome? a field experiment on password policy. *Computers & Security*, 88:101617.
- vanhauser the (2014). Thc-hydra. <https://github.com/vanhauser-thc/thc-hydra>. (last accessed on 22/11/22).
- Yisa, V., Baba, M., and Olaniyi, E. (2016). A review of top open source password cracking tools. International Conference on Information and Communication Technology and Its Applications.
- Yu, F. and Yin, H. (2021). Password cracking of pdf 2.0 documents on gpu. In 2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS), pages 721–725. IEEE.
- Zion3R (2018). hash-identifier. <https://github.com/blackploit/hash-identifier>. (last accessed on 05/12/22).