5-2018

Artificial Intelligence-Based Password Brute Force Attacks

Khoa Trieu
Fontbonne University, trieuk@fontbonne.edu

Yi Yang
Fontbonne University, yyang@fontbonne.edu

Artificial Intelligence-Based Password Brute Force Attacks

Khoa Trieu
Fontbonne University
St. Louis, MO
USA
trieuk@fontbonne.edu

Yi Yang
Fontbonne University
St. Louis, MO
USA
yyang@fontbonne.edu

ABSTRACT

Brute force attack is a usual way to crack passwords based on a crafted dictionary. Traditionally, this dictionary is constructed using an existing pool, random words, meaningful words from a public website, or prior passwords, which makes the brute force attack take long time and consume a lot of resources. Lately, two interdisciplinary fields of Cyber Security and Artificial Intelligence (AI) converge together. On one hand, researchers apply artificial intelligence especially machine learning or pattern recognition to make offenses and defenses in cyber security smarter. On the other hand, cyber security technologies are used to protect artificial intelligence algorithms/modules, making them safer. Under this umbrella, we begin to think about next-generation password brute force attacks based on artificial intelligence. We propose to use an open-source machine learning algorithm called Torch-rnn, which is available from GitHub, to generate new potential passwords following a similar pattern based on prior passwords and insert them into the brute force dictionary in real time. Hence, our password brute force attacks become smarter and more efficient. Our experimental studies indicate that AI - based password brute force attacks have significantly higher success/hit rates to crack the correct passwords, compared with non AI - based (or traditional) password brute force attacks. In this paper, we also propose defensive strategies to protect our passwords against this new-generation and smarter AI-based password brute force attacks.

Keywords

Cyber security, artificial intelligence, brute force attacks, passwords, dictionary.

INTRODUCTION

Password cracking is a usual way to break a computer system's authentication mechanism. Modern computer systems, such as Linux and Unix, use hash algorithms, e.g. MD5 and SHA-1 (or SHA-2), to compute hash values from original passwords, and store those hash values locally in the passwords database. This can prevent the attacker from directly retrieving password plaintexts. However, the attackers can brute force the hash values of all the possible passwords, by using e.g. a rainbow table. Once they find a match, they can break into the system using the username and corresponding password. The introduction of salt (e.g., from the current system timestamp) into the password hashing mechanisms significantly improves the security of computer systems' authentications. So far, password brute force attacks based on a crafted dictionary are still an effective way to break into a computer system (Javed and Paxson, 2013). The speed of password cracking from a brute force attack largely depends on how we construct and update the dictionary. For example, John the Ripper (John 2018) is a popular tool to crack passwords based on brute force. The performance of this tool depends on the quality of dictionary.

Lately, two interdisciplinary fields of Cyber Security and Artificial Intelligence (AI) merge together (Nelson et al., 2013; Akusok et al., 2017). On one hand, researchers apply artificial intelligence especially machine learning (Patel, 2010) and pattern recognition to make cyber security offenses and defenses smarter. On the other hand, cyber security technologies are used to protect artificial intelligence algorithms/modules and make them safer (Carrara et al., 2017; Wang and Zhu, 2017). Under this umbrella, we begin to think about how to make password brute force attacks smarter and more efficient/effective, by constructing the attacking dictionary in a more intelligent way. We propose to use artificial intelligence especially machine learning to construct the dictionary in a smarter way, which can significantly improve the attacking performance by increasing the hit/success rates of password brute force attacks.

Our experimental studies are based on an open-source machine learning algorithm called Torch-rnn (Torch 2018), which is available from GitHub, to generate new potential passwords following a similar pattern as prior passwords and insert them into the brute force dictionary in real time. In this way, our password brute force attacks become smarter and more efficient. Our performance evaluation results show that AI – based password brute force attacks have significantly higher success/hit rates to crack the correct passwords, compared with non AI – based (or traditional) password brute force attacks. In this

paper, we also propose defensive strategies to protect our passwords against this new-generation smarter AI-based password brute force attacks.

RELATED WORK

Computer authentication systems

Authentication is the first defensive line in computer systems to defend against an intruder or attacker. Authentication methods are normally based on something you know (e.g., your social security number), something you have (e.g., keys to open a door), and something you are (e.g. fingerprints). Passwords fall in the category of something you know. Choosing a strong password for a computer system is very important to prevent the break-in from an attacker. A strong password means a long random sequence with a combination of special characters, numbers, and upper-case letters. Also, in practice, we should not use the same password for different computer systems, such as university emails, bank accounts, and social medias. However, stronger security may influence the usability of password systems, because stronger passwords tend to be harder for people to remember. Like many other computer security systems, there is a tradeoff between security, usability, and sometimes performance.

Originally, usernames and corresponding passwords are stored in computers in plaintexts for authentication purposes. In this case, once the database of usernames and passwords is cracked, the attacker is able to obtain all the users' login credentials. This is a disaster. Later, researchers apply hash algorithms, such as MD5 and SHA-2, to hash all the passwords before storing them into the local database. Even if one day this local database might be released, without the keys, the attacker cannot figure out password plaintexts assuming good hash functions are one-way. However, tools based on rainbow tables, such as RainbowCrack (Rainbow 2018), can still break passwords efficiently based on a precomputed hash chain.

The introduction of salt, a random value generated from e.g. the current system timestamp, can make the situation much better. Salt-based password creation and verification are illustrated as follows in Figure 1 (Practice 2018). Basically, we introduce a random value called salt into the calculations of hash values, which makes rainbow tables based on precomputed hash values useless. Authentication such as multi-factor mechanisms is still an active research field in computer and network security.

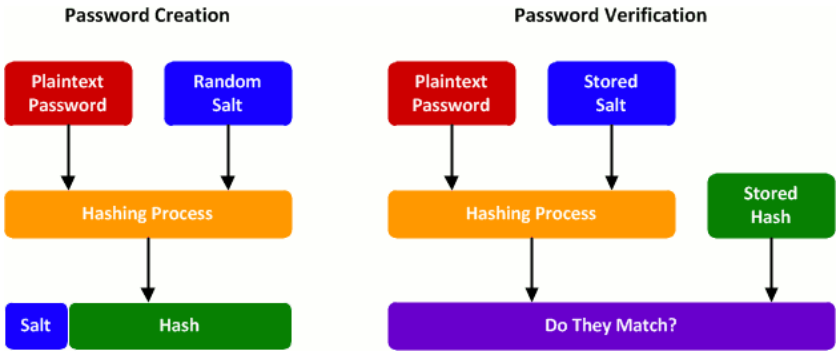

Figure 1. Salt-Based Passwords Creation and Verification

Password cracking mechanisms

Passwords cracking mechanisms could be broadly categorized into two types: the first type is brute force attack; the second one is cryptographic analysis. Brute force attack can crack the correct passwords eventually after it tries all the possible combinations. However, a good dictionary can boost the speed of brute force attacks. Cryptographic analysis, such as timing based analysis and side/convert channel analysis, can crack passwords in a more efficient way based on more information available, such as timing data/records and some available (plaintext, ciphertext) pairs. This paper explores artificial intelligence to improve the performance of brute force attacks by self-learning and constructing a more intelligent dictionary and applying it in the brute force attacks in real time. Experimental studies show that this proposed technique can significantly improve the success/hit rates of password brute force attacks.

TRADITIONAL PASSWORD BRUTE FORCE ATTACKS

Dictionary-based brute force attacks

The following flow chart (in Figure 2) illustrates the basic idea of dictionary-based password brute force attacks.

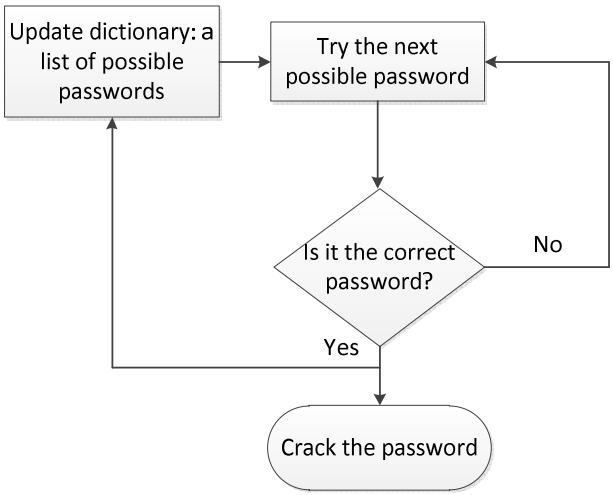

Figure 2. Dictionary-Based Password Brute Force Attacks

Rainbow tables

A rainbow table is a precomputed table of hash chains for reversing cryptographic hash functions, usually for cracking password hashes. A simple example of rainbow table with three reduction functions is shown in Figure 3 (Rainbow, 2018). From here we can see that precomputed hash values are chained together in an efficient way. Once we find a match of hash values, then the previous plaintext in the hash chain is the correct password.

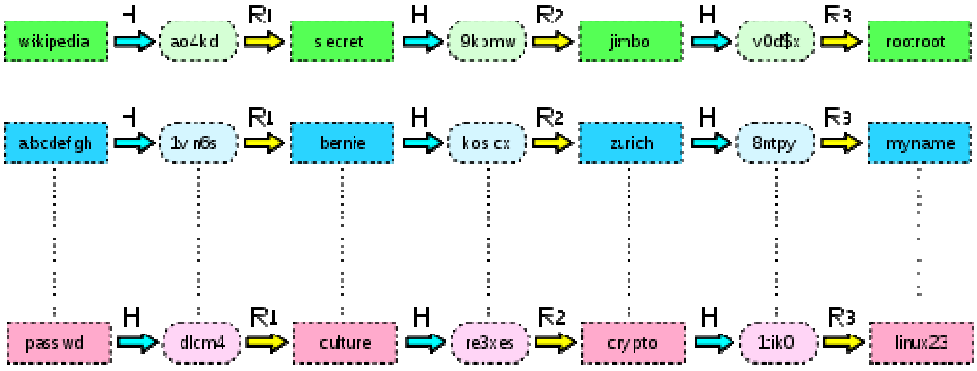

Figure 3. An Example of Rainbow Table With Three Reduction Functions

ARTIFICIAL INTELLIGENCE-BASED PASSWORD BRUTE FORCE ATTACKS

An overview

We first present an overview of the AI-based algorithm through a flowchart. This flowchart is shown in Figure 4. Then, we discuss more details about our implementation with three steps: training AI, check points, and applying the improved dictionary in the brute force attacks.

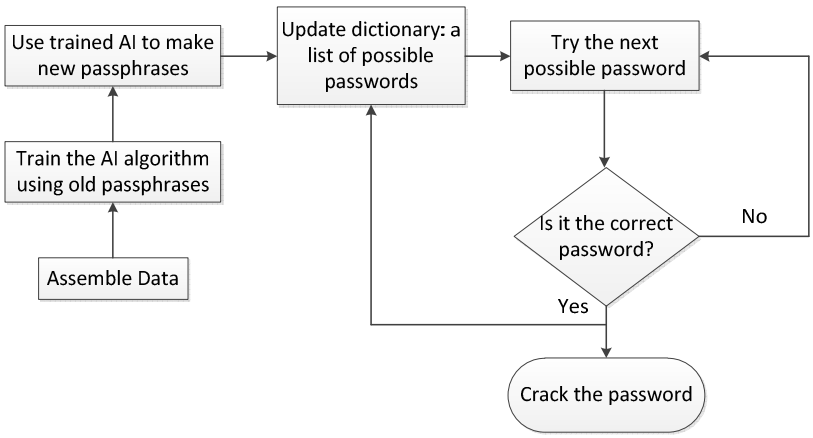

Figure 4. An Overview of AI-Based Password Brute Force Attacks

The detailed process

Training AI

We prepare the data using a python script file called preprocess.py.

```
(.env) ace@ace-VirtualBox:~/torch/torch-rnn$ python scripts/preprocess.py --inp
ut_txt sh_Large_Corrupt.txt --output_h5 sh_V_Corrupt.h5 --output_json sh_V_Corr
upt.json
Total vocabulary size: 60
Total tokens in file: 1155392
  Training size: 924314
  Val size: 115539
  Test size: 115539
Using dtype  <type 'numpy.uint8'>
```

Check points

The AI training creates multiple check point files to make sure that AI learns correct information from past passphrases and it is able to generate accurate new information.

```
checkpoint_1000.json  checkpoint_4000.t7    checkpoint_6000.json
checkpoint_1000.t7    checkpoint_5000.json  checkpoint_6000.t7
checkpoint_2000.json  checkpoint_5000.t7    checkpoint_7000.json
checkpoint_2000.t7    checkpoint_500.json   checkpoint_7000.t7
checkpoint_3000.json  checkpoint_500.t7     checkpoint_8000.json
checkpoint_3000.t7    checkpoint_50.json    checkpoint_8000.t7
checkpoint_4000.json  checkpoint_50.t7
```

Apply the improved dictionary in brute force

The last step is to apply the improved dictionary in password brute force attacks. We develop a password brute force attack GUI (Graphic User Interface) using C# to demonstrate the password brute force cracking procedure, as shown in Figure 5.

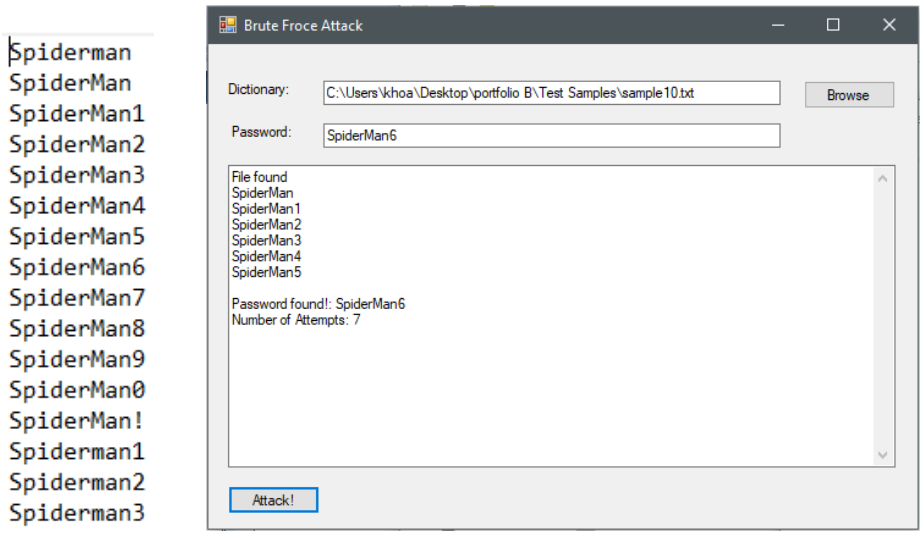

Figure 5. An AI-Based Password Brute Force Attacks GUI

THE COMPARISON OF AI-BASED AND NON-AI BASED ALGORITHMS BY EXPERIMENTAL STUDIES

Evaluation setup

We implement AI-based password brute force attacks based on Torch-rnn benchmarks. For comparison purposes, we also implement non-AI based (or traditional) password brute force attacks. We compare the performance of these two different algorithms in terms of success/hit rates of correct passwords. We run the comparisons for different lengths of the dictionary (i.e. the number of words in the dictionary): 50, 100, 250, 500, 750, and 1000. For each trial, we calculate the average of 100 tries.

Evaluation results

The evaluation results are shown in Table 1 and Figure 6. From here we can see that compared with traditional brute force algorithm the AI-based algorithm can significantly increase the success or hit rates of correct passwords for all the six trials that we tried.

Table 1. Comparison of AI-Based Algorithm and Traditional Algorithm Through Experimental Studies

# of Words in the Dictionary	Success Rates of AI-Based Algorithm	Success Rates of Traditional Algorithm
50	24%	2%
100	38%	3%
250	35%	13%
500	44%	12%
750	56%	16%
1000	57%	24%

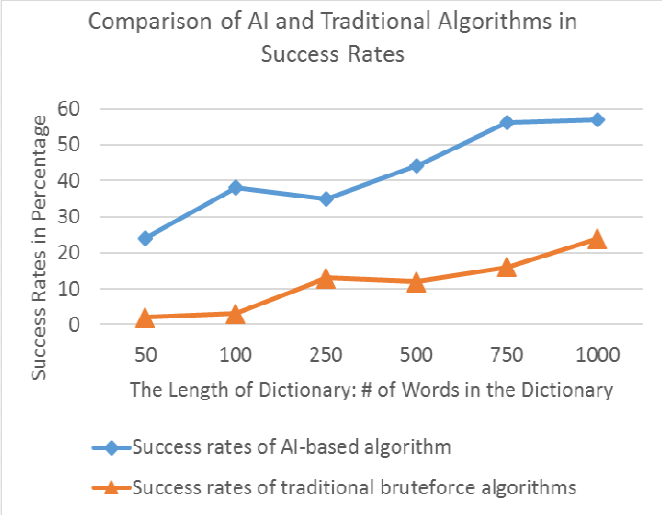

Figure 6. Comparison of AI-Based and Traditional Algorithms in Success/Hit Rates

DEFENSIVE STRATEGIES AGAINST THE NEW-GENERATION AI-BASED PASSWORD BRUTE FORCE ATTACKS

We can apply AI-based password brute force algorithms to prevent users from choosing poor new passwords similar to old ones. In practice, users should choose stronger passwords by using random combinations of special characters, numbers, and upper-case letters. Furthermore, single-factor authentication purely based on passwords might be easily hacked by innovative technologies such as the AI-based algorithms. So, we should develop multi-factor authentications based on combinations of different verification factors, such as text messages on smartphones, emails, and passwords.

CONCLUSION AND FUTURE WORK

In this paper, we develop a new password cracking technique based on artificial intelligence. Performance evaluation results indicate that this new technique has significantly higher hit/success rates of correct passwords, compared with traditional or non-AI based brute force attacks. Our future work will focus on applying AI-based algorithms to prevent users from choosing poor passwords and also developing a defensive framework to prevent AI-based brute force algorithms from being effective.

ACKNOWLEDGMENTS

We thank the Department of Mathematics and Computer Science in the College of Arts and Sciences at Fontbonne University to sponsor the research of this project and also the conference registration.

REFERENCES

1. John the Ripper (2018). http://www.openwall.com/john/.
2. Fabio Carrara, Fabrizio Falchi, Roberto Caldelli, Giuseppe Amato, Roberta Funmarola, and Rudy Becarelli (2017) Detecting Adversarial Example Attacks to Deep Neural Networks, *CBMI'17 (Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing),* Article No. 38.
3. Weiyu Wang and Quanyan Zhu (2017) On the Detection of Adversarial Attacks against Deep Neural Networks, *SafeConfig '17 (Proceedings of the 2017 Workshop on Automated Decision Making for Active Cyber Defense)*, Pages 27-30.
4. Anton Akusok, Emil Eirola, Kaj-Mikael Bjork, Yoan Miche, Hans Johnson, and Amaury Lendasse (2017) Brute-force Missing Data Extreme Learning Machine for Predicting Huntington's Disease, *PETRA'17 (Proceedings of the 10th International Conference on Pervasive Technologies Related to Assistive Environments)*, Pages 189-192.
5. Blaine Nelson, Christos Dimitrakakis, and Elaine Shi (2013) Summary/Overview for Artificial Intelligence and Security, *AISec '13 (Proceedings of the 2013 ACM SIGSAC conference on computer and communications security)*, Pages 1483-1484.

6. Kayur Patel (2010) Lowering the Barrier to Applying Machine Learning, *CHI EA'10 (CHI'10 Extended Abstracts on Human Factors in Computing Systems)*, Pages 2907-2910.
7. Mobin Javed and Vern Paxson (2013) Detecting Stealthy Distributed SSH Brute-Forcing, *CCS'13 (Proceedings of the 2013 ACM SIGSAC conference on computer and communications security)*, Pages 85-96.
8. Torch recurrent neural networks (2018). https://github.com/torch/rnn.
9. RainbowCrack (2018) http://project-rainbowcrack.com/.
10. Worst and best practices to secure password storage (2018). https://blog.conviso.com.br/worst-and-best-practices-for-secure-password-storage/.
11. Rainbow tables (2018) https://en.wikipedia.org/wiki/Rainbow_table.