Ayo Tool

An Advanced Password Cracking Technique If Dictionary and Brute Force Attacks Have Failed

Alin Boncioaga | orcid.org/0000-0002-8871-6235
Bucharest | Romania | 2019
alin.boncioaga@gmail.com
Code on Github : https://goo.gl/d2GgSY

DOI: 10.29322/IJSRP.9.03.2019.p8748
http://dx.doi.org/10.29322/IJSRP.9.03.2019.p8748

Abstract

Ayo is a POC tool that's designed to make password cracking techniques easier. It should only be used if the following strategies have failed:

- dictionary attack (including rainbow tables).
- brute force attack (including masks).
- password profiling (including Cewl, Wyd, and Cupp).

Ayo is a powerful password generation tool that is capable of learning any word order derived from PDF books. Basically, it can process PDF books in a matter of minutes if not seconds and will learn the normal flow of words in a sentence to generate language-specific possibilities. Masks help to generate custom passwords with special characters and/or numbers.

Keywords

Password profiling, big data password profiling.

1. Introduction

For the purpose of illustration, a real-world pen test scenario is used.

A contract has been awarded for pen testing a Spanish legal firm. However, due to the rules of engagement, social engineering and other techniques cannot be considered. The team has access to the company's premises and has managed to extract some WPA hashes with a simple airodump–ng scan. However, after several attempts, the team is not able to crack any other hashes using the most commonly available methods, such as dictionary attack, brute force attack, Cewl, Wyd and Cupp.

Using the keywords "Spanish company" and "company in the legal sector," domain-specific passwords are generated in the target language using Ayo. A number of legal PDF books written in the Spanish language are fed to the tool for this case.

Initially, this generates a number of questions:

- Is this feasible?
- Would the result crack any actual hashes?
- Would this generate an overwhelming number of passwords?

2. Data Structures

Once the PDF books have been processed, the word order is retained in a tool-specific data structure. This resembles a neural network-like web.

2.1 Example

The word "blue" is used in this example. In a typical English sentence, this would be followed by a word such as sky, car, or eyes and not by words such as cow or dog.

Learn Mode will store all the words in the following structure:

Dictionary<string, Dictionary<string, int>>

Each example will list the words that have occurred after it, plus the number of occurrences. This structure will be written by **Learn Mode** in JSON format and placed in a separate file.

2.2 Scalability

This is a Dictionary within Dictionary software design

Search Operation O(1). However, there are two main issues related to memory object size and output file size, and further development is needed to improve scalability.

3. Usage

Ayo will run in three distinct modes. They are designed to be run in sequence as the output of each mode becomes the input for the following one.

While any tool in the marketplace can be used for the third functionality mode, **Learn Mode** and **Generate Mode** specifically introduce new ideas to the field of academic cryptography.

3.1 Learn Mode

The objective is to learn the sequence of words from PDF books in order to generate smart possible passwords. The output of this mode is a JSON extension file that holds the word structure. The data is stored in such a way that it generates constant complexity (O(1)) within the search operation.

The full command examples provided in the help menu have been tested and found to be 100% functional. However, due to time restrictions and modest financial resources, there remain many untested workflows and possible bugs within the current POC implementation.

3.1.1 Clean Operation

- - clean 30 eliminates 30% of the words with the least connections. The clean operation O(n*n) is only performed once and is therefore negligible.

3.1.2 Example

Ayo.exe – learn – path % path_to_folder % -- write % path_to_folder% --clean 30

3.2 Generate Mode

This generates a password based on the JSON extension file created during the previous step and a mask.

that's been chosen for its speed within

3.2.1 Mask Options

x – word.
c – capitalise the first letter of next word.
C – capitalise all letters of the next word.
n – numbers.
s – special character.

3.2.2 Example

example: ayo.exe --generate --json % path_to_json% -- write %path_to_txt_file% -wnsw --nr 10000

4. User Interface

Ayo generates a simple Windows CLI. For example, ayo -- help will generate the following:

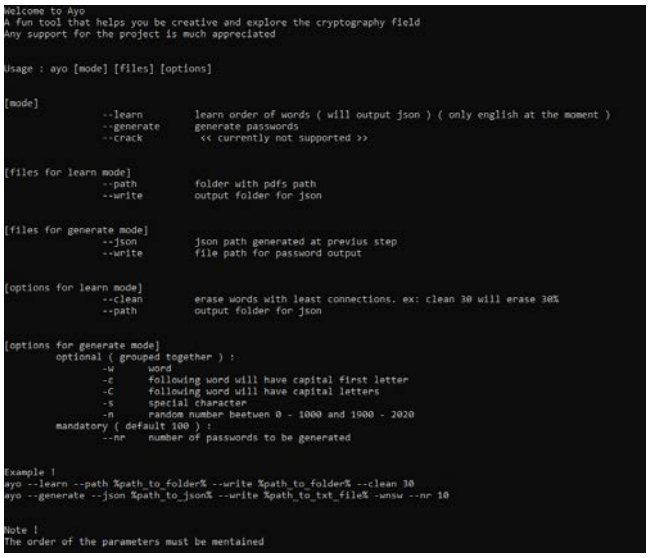

Figure 1

5. Results

It's important to remember that this is a simple tool using clean and basic code. The main benefits of Ayo can be found in what it brings to the password profiling niche.

5.1 Adjustable Parameters

To obtain the best results, it's important to focus on two parameters in detail – " --clean" from **Learn Mode** and " --nr" from **Generate Mode**.

" --nr" refers to the number of passwords required.

" --clean" will erase the words with the least connections to leave only those that have been encountered most often in linguistic situations using the chosen domain/language/PDF Book.

5.2 Big Data Password Profiling

Typically, password profiling tools generate a string based on a word combination, such as dog name, date of birth, mother's maiden name, friend's name, favourite sport, and so on. This tool allows the password profiling niche to work with large datasets. It will always generate consistent results. Nearly all the generated strings are good candidates of being able to crack the hash.

5.3 Examples of Generate Mode

The following CLI commands exclude " --json" and " --write" arguments.

5.2.1 --generate -cwcw --nr 3

Result 1: EncodeArbitrary
Result 2: RulesApply
Result 3: ReusableProofs

5.2.1 --generate -cwncwncw --nr 3

Result 1: Added886To839Catch
Result 2: Nearzero390By189Yonatan
Result 3: Solutions668To580Spend

5.2.2 --generate -cwscwnCw --nr 3

Result 1: Use"Two37WAYS
Result 2: Similar!To85ME
Result 3: Sign%The781FORM

6. Limitations

At the moment, the tool only supports the **Learn Mode** and **Generate Mode** apps. It is written in C# as a Windows tool and works in English only for WPA hashes.

Ayo tool takes this a step further toward a big data password profiling environment.

However, support for any language can be added to the modular architecture of the tool. This can be achieved by replacing language specific characters with ASCII ones. For example, replace 'ù' with 'u'.

Note that **Cracking Mode** is not developed yet.

The tool is a POC. While comprehensive tests have been performed, if more than 200 PDF books (with more than 500 pages each) are loaded, users may encounter bugs.

7. Future Plans

Ayo has solid architecture and design patterns. Consequently, its tools can easily be maintained or extended. Future plans include:

- **Learn Mode** to process PDFs in multiple languages.
- **Crack Mode** to work with multiple hashing/encryption algorithms.
- Linux CLI./GUI
- Windows GUI

8. Artificial Intelligence in Password Profiling

Ayo is a very flexible and powerful tool. As it simply relies upon PDF files for input, any number of subjects, areas of interest or languages can be introduced. With "clean operation," many of the more unusual word combinations can be filtered out.

While it would have been possible to add any number of additional or more complex features, simplicity was deemed to be critical and the tool's most important asset. Remember that the resulting JSON structure holds the sentence structure learned by the tool. From this point, knowledge gathered from the PDF books can be used to construct possible passwords.

9. Conclusion

This tool successfully demonstrates that password cracking can extend beyond dictionary and brute force attacks. Even though tools like Cewl, Wyd and Cupp have proven that custom targeted password generation is very useful, the

About the Author :

Alin Boncioaga is a passionate software developer from Romania, active in the cybersecurity space and has a few companies of his own. Open Source enthusiast.

Acknowledgments :

A warm ‘Thank You’ to Vlad Monescu and Ioana Cristina Plajer, Transilvania University Romania, for they passion for algorithms and teaching.

References:

[1] S.O.L.I.D: The First 5 Principles of Object Oriented Design by Robert Martin, 2000.

[2] WHY PROGRAMS FAIL: A Guide to Systematic Debugging by Andreas Zeller, 2009.

[3] Design Principles and Design Patterns by Robert Martin https://fi.ort.edu.uy/innovaportal/file/2032/1/design_principles.pdf, 2000.

[4] Handbook of Applied Cryptography (Discrete Mathematics and Its Applications) by Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, 2001.

[5] Aircrack-ng by Ronald Cohn Jesse Russell, 2012.

[6] Mastering Kali Linux for Advanced Penetration

Testing: Secure your network with Kali Linux - the ultimate white hat hackers' toolkit, 2nd Edition by Vijay Kumar Velu, 2017.

[7] Wireshark Network Analysis: The Official Wireshark Certified Network Analyst Study Guide by Laura Chappell, 2010.

[8] Seven Deadliest Wireless Technologies Attacks by Brad Haines, 2010.