Vol. 04, No. 2,(2020),063–071 , **ISSN:2581-3242**

IJMLNCE JOURNAL

International Journal of Machine Learning and Networked Collaborative Engineering

Journal Homepage: http://www.mlnce.net/home/index.html

DOI : https://doi.org/10.30991/IJMLNCE.2020v04i02.002

Utilize Machine Learning Methods to Detect Plaintext Passwords

[a]Nada Al-Noaimi, [b]Abdullah Al-Turaifi, [c]Sireen Babateen

[a] Cybersecurity Expert, Department of Information Protection, Saudi Aramco, Saudi Arabia, noaimine@hotmail.com

[b] Cybersecurity Expert, Department of Information Protection, Saudi Aramco, Saudi Arabia, aat700@gmail.com

[c] Application Development Expert, Department of Corporate Applications, Saudi Aramco, Saudi Arabia,sireen.babateen@gmail.com

Abstract

Every company is a target today, no matter the type of business it does. Hackers and cybercriminals are after data which they can monetize in many ways. Being proactive and having a defensive and protective plan in place — such as evaluating and assessing IT security — is an excellent recipe for avoiding data breaches, and consequently, business disasters. Passwords are the most popular authentication method, mainly because they are easy to implement, require no special hardware or software, and are familiar to users and developers. Unfortunately, most users store their sensitive information or credentials in plaintext that might be accessible to attackers. Since the information is not encrypted and stored or transferred in cleartext, attackers will quickly read it. Keeping a plaintext password in a configuration file allows anyone to read the file access to the password-protected resource. Developers sometimes believe that they cannot defend the application from someone who has access to the configuration, but this attitude makes an attacker's job easier. Useful password management guidelines require that a password must never be stored in plaintext. The question is, why not utilize a machine learning platform (MLP) that can be trained to search text in a computer resource, detect a string of plaintext characters, and analyze the string of characters to predict or detect a plaintext password on a computer resource asset.

Keywords

Machine Learning,

Plaintext,

Passwords,

Security Method,

Information Security

1. Introduction

Plaintext passwords can be stored anywhere on a computer network, including on a computer resource asset, such as, for example, a file (for example, a configuration file). A router, a switch, a computer, a server, a database, or source code; the solution can be arranged to target computer resource assets on the network and search those assets.

The MLP will be able to detect a plaintext password in a character string by analyzing plaintext character strings for typical password complexity, such as, for example, including at least one uppercase letter, lowercase letter, number, unique character, and text length (for example, minimum of eight characters). Then check the similarity of the character string against a database comprising passwords, including, for example, passwords that were previously found or identified by the solution or passwords that were input or loaded into the database from a list, table, record, file, or a computer resource that can input passwords to the database. The MLP will also predict a level of certainty that a character string includes a password and output a confidence score based on the expected level of certainty. Finally, the MLP will categorize the confidence score in any number of prediction certainty levels, including, for example, three levels – high, medium, or low.

2. Proposed Model

The platform will identify appropriate remediation based on the confidence score or prediction certainty level. The remediation can include, for example, effectuating encryption on the computer resource asset where a confirmed plaintext password is located, effectuating password verification, or classifying and labeling the plaintext character string as a false positive. It will also ensure the security and integrity of a computer resource asset or a computer network system to which the computer resource asset is connected by detecting plaintext passwords and encrypting the computer resource or extracting the plaintext passwords replacing them with corresponding ciphertext tokens.

Fig. 1 shows a non-limiting embodiment of a computer network 10 having a plurality of nodes N1, N2, N21, N22, N23, N3, N31, N32, N33, N4, N41, N42, N43, N44, and N5 (collectively or individually referred to as a node "N"). The computer network ten can include, for example, tens, hundreds, thousands, millions, or more nodes N, any of which can consist of one or more computing resource assets. Each node N can include a location identifier that can identify the node's physical or virtual address on the computer network 10. The node location identifier can include, for example, an Internet Protocol (IP) address, a Media Access Control (MAC) address, an Ethernet Hardware Address (EHA), hardware address, adapter address, physical address, or virtual address. Each node N can include one or more computing resource assets (CRA) 20. Each CRA 20 can consist of a computer resource 22, a computing device 24, or a communicating device 26, for example, as seen for node N31. The cybersecurity solution can search text in one or more computer resources 22 on a target computer resource asset, such as, for example, the CRA 20 located at node N31 on the computer network 10.

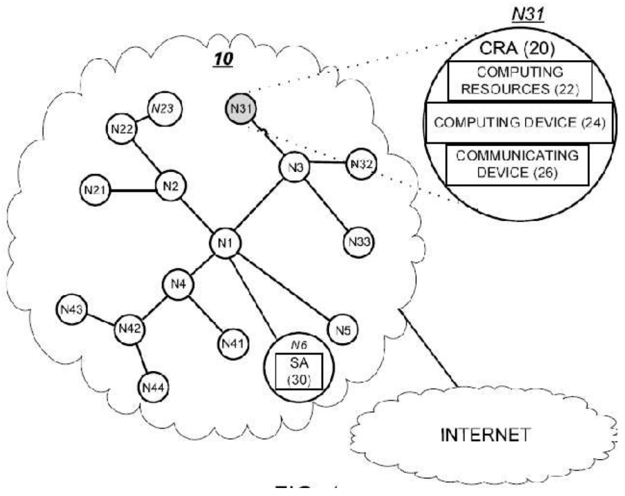

Figure 1

The computer network ten can include an embodiment of the cybersecurity solution, according to the disclosure principles. The answer can consist of a security appliance 30. The security appliance 30 can be located at one or more nodes N on the computer network 10, such as, for example, at node N6. The security appliance 30 can be located outside of the computer network 10, such as, for instance, on a cloud network (not shown) or the Internet. The security appliance 30 can include a cybersecurity system 100.

FIG. 2 shows a non-limiting embodiment of the cybersecurity system 100. The system 100 can be configured to implement the various aspects of the solution. The system 100 can include a processor 110, a storage 120, a network interface 130, an input-output (IO) interface 140, a driver suite 150, a password detector suite 160, a model training and tuning (MT&T) unit 170, an encryption unit 180 and a remediation unit 190. The encryption unit 180 can be included in the remediation unit 190, as seen in FIG. 2, or provided as a separate computer resource asset. The system 100 can include a bus 105, which can be connected to any or all of the computer resource assets 110 to 190 by a communication link.

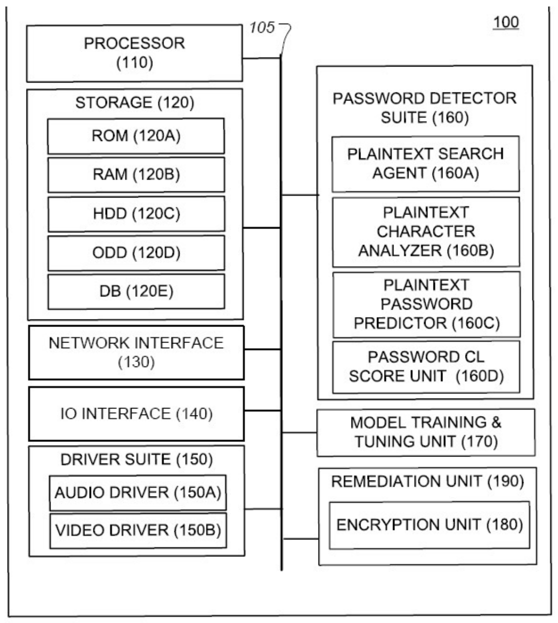

Figure 2

The machine learning (ML) model can be arranged to receive plaintext as input, including a string of characters, extract features from the plaintext, classify the extracted features, cluster or group results, and predict a plaintext password in the characters' line based on an analysis of the plaintext. The ML model can be built, trained, or tuned by the MT&T unit 170. The MT&T unit 170 can instruct the ML model using annotated historical data, including the training dataset stored in the DB 120E, and tune the ML model using the testing dataset stored in the DB 120E. The MT&T unit 170 can adjust the ML model during operation of the system 100 by updating parametric values in the ML model based on, for example, outputs from a plaintext password remediation process 200.

The password detector suite 160 can include one or more computing resource assets, including a plaintext search agent 160A, a plaintext character analyzer160B, a plaintext password predictor 160C, and a password certainty-level (CL) score unit 160D. Anyone or more of the computing resource assets, including 160A, 160B, 160C, or 160D, can include an MLP. In a non-limiting embodiment of the system 100, the password detector suite 160 is constructed as an MLP, in which a plurality of neural network layers is built to operate as the plaintext search agent 160A, plaintext character analyzer160B, plaintext password predictor 160C, or password CL score unit 160D.

The plaintext search agent 160A can be arranged to interact with, for example, the processor 110, storage 120 or network interface 140, to traverse computer resource assets on the computer network 10 (shown in FIG. 1). The plaintext search agent 160A can be arranged to cross computer resources 22 (such as, for example, configuration files) on the CRA 20 (shown in FIG. 1) and search for plaintext. The plaintext search agent 160A can include a crawler, spider, or bot. The plaintext search agent 160A can be arranged to index strings of characters or copy, deconstruct or extract strings of characters comprising plaintext from the computer resource 22 and input the plaintext character strings to the plaintext character analyzer160B. The plaintext search agent 160A can be arranged to interact with the storage 120 to store plaintext characters in, for example, the DB 120E (shown in FIG. 1). The DB 120E can store metadata corresponding to the computer resource 22 from which the plaintext character strings were copied, deconstructed or extracted, including, for example, location of the plaintext character string in the computer resource 22 and time of copying, deconstruction or extraction.

The plaintext search agent 160A can be arranged to receive as input a list of plaintext passwords from the DB 120E or an external source, and search text on each computer resource 22 on the CRA 20 (shown in FIG. 1) to find a matching plaintext password for any or more of the passwords on the input list.

The plaintext search agent 160A can be arranged to preprocess strings of plaintext characters. The preprocessing can include tokenization or another technique to eliminate unconcerned differences, such as, for example, non-password text such as common use words. The preprocessing can include sizing and normalization of the plaintext character strings. The plaintext search agent 160A can be arranged to input the plaintext character strings to the plaintext character analyzer160B.

3. Classification Methodology

The plaintext character analyzer160B can be arranged to preprocess the strings of plaintext characters.The plaintext character analyzer160B can be set to analyze each character's line, including each character in a plaintext character string, and classify each character or the plaintext character string. The plaintext analyzer160B can include feature extraction and vector classification. The plaintext analyzer160B can be arranged to extract features from the strings of characters based on, for example, the factors included in table 300 (shown in FIG. 4). The plaintext analyzer160B can output a classification for each character or plaintext character string to the plaintext password predictor 160C. The plaintext analyzer160B can be arranged to annotate each string of characters determined to include a plaintext password with a label. The plaintext analyzer160B can interact with the MT&T unit 170 and output confirmed plaintext passwords with corresponding labels and classification data, to the MT&T unit 170, which in turn can be used by the MT&T unit 170 to build or update the training dataset or testing dataset.

Criteria	Yes/No	Granted Points
Uppercase Letters	No	0
Lowercase Letters	Yes	1
Text Length (minimum of 8 characters)	Yes	1
Special characters	Yes	1

The plaintext character analyzer160B can be arranged to analyze each character in a plaintext character string according to a plurality of factors to determine whether the character includes, for example, an uppercase letter, a lowercase letter, a number, or a unique character. The total number of characters in the plaintext character string can be counted. The factors can include, for example, the factors in table 300 (shown in FIG. 4).

The plaintext character analyzer160B includes a CNN or DCNN, in which case, the plaintext character analyzer160B can format the plaintext character strings into matrices and filter each matrix using at least one convolution filter by sliding the convolution filter across (for example, as a function of time) each matrix to compute dot products and detect patterns. The plaintext character analyzer160B can slide and apply multiple convolution filters across multiple matrices of plaintext character data to extract a plurality of feature maps for the plaintext character data. Once the feature maps are extracted, the feature maps can be moved to one or more rectified linear unit layers (ReLUs) in the neural network to locate the features. After the features are located, the corrected feature maps can be moved to one or more pooling layers to down-sample and reduce each feature map's dimensionality. The down-sampled data can be output as a onedimensional data array or multidimensional data arrays from the pooling layers and flattened (or converted) into single continuous linear vectors that can be forwarded to the fully connected layer. The flattened matrices from the pooling layer can be fed as inputs to a fully connected neural network layer, auto-encoding the feature data from the feature extraction, and classifying the character data. The plaintext character analyzer 160B can include a fully connected layer that contains a plurality of hidden layers and an output layer. The output layer can output character classification data to the plaintext password predictor 160C.

The plaintext password predictor 160C can be arranged to receive the output from the plaintext character analyzer160B, including character classification data, and predict a probability that a plaintext character string includes a plaintext password. The plaintext password predictor 160C can generate a confidence score for character or plaintext character string, including the likelihood that a bounding box consists of a plaintext password or a plaintext password character. The plaintext password predictor 160C can interact with the plaintext character analyzer160B and perform bounding box classification, refinement, or scoring based on the characters in the plaintext character string. The confidence score can have a value ranging from, for example, 0 to 5, 0 to 10, or 0% to 100%, with 100% being a confirmed plaintext password and 0% being confirmed non-password plaintext.

The confidence score can be output by the plaintext password predictor 160C to the password certainty-level (CL) score unit 160D. The password CL score unit 160D can categorize the analyzed plaintext character string into one of three certainty levels: a high certainty level, a medium certainty level or a low certainty level. The plaintext character string can be classified into more or less than three certainty levels. In a non-limiting example, the high certainty level includes plaintext character strings for which the confidence score value is greater than or equal to 70% (up to 100%); the medium certainty level includes plaintext character strings for which the confidence score value is greater than or equal to 30% but less than 70%; and the low level includes plaintext character strings for which the confidence score value is less than 30%.

In the password detector suite 160, the plaintext character analyzer160B, plaintext password predictor 160C, and password CL score unit 160D can be implemented in a non-limiting embodiment using one or more CNNs having several convolutional/pooling layers (for example, 1 or 2 convolutional/pooling layers) and a single fully connected layer, or it can be implemented using a DCNN having many convolutional/pooling layers (for example, 10 or more layers) followed by multiple fully connected layers (for example, two or more fully connected layers). The password detector suite 160 can be arranged to develop or operate the ML model based on both input and output data (supervised learning model) or group and interpret data based only on input data (unsupervised learning model).

4. Results

The MT&T unit 170 can be arranged to interact with the password detector suite 160, train the ML model using the training dataset, and tune the ML model using the testing dataset or outputs from the process 200 (shown in FIGS. 3A and 3B). The MT&T unit 170 can be arranged to interact with an operator via, for example, the IO interface 140 and receive annotations or instructions from the operator to build or update the training dataset. The ML model can be tested using the testing dataset, and the results can be fed back to tune the ML model's parametric values. Once the ML model is trained, the MT&T unit 170 can be arranged to continuously tune the ML model by updating the ML model's parametric values, based on the outputs from the process 200 (shown in FIGS. 3A and 3B).

The encryption unit 180 can be arranged to convert a plaintext password into a corresponding ciphertext toke or to convert all (or less than all) plaintext in the computer resource containing the plaintext password into ciphertext. In a non-limiting example, the encryption unit 180 can encrypt all plaintext in a configuration file into ciphertext. The encryption unit 180 can include one or more encryption or hashing algorithms. The encryption unit 180 can include, for example, a password-based key derivation function (for example, PBKDF1, PBKDF2 or scrypt), a password hashing function (for example, bcrypt), or any encoding technique that can protect and defend against attempts or attacks (for example, brute-force attacks) to decrypt or decode the plaintext password from the ciphertext.

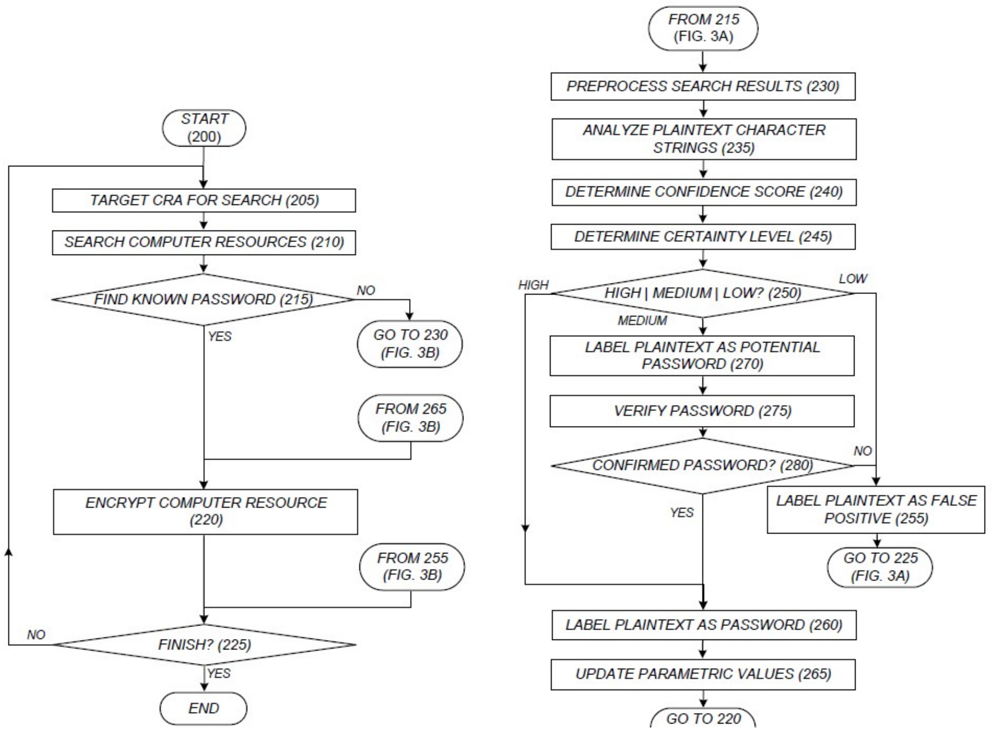
START (200)
TARGET CRA FOR SEARCH (205)
SEARCH COMPUTER RESOURCES (210)
FIND KNOWN PASSWORD (215)
NO
YES
GO TO 230 (FIG. 3B)
FROM 265 (FIG. 3B)
ENCRYPT COMPUTER RESOURCE (220)
FROM 255 (FIG. 3B)
NO
FINISH? (225)
YES
END
FROM 215 (FIG. 3A)
PREPROCESS SEARCH RESULTS (230)
ANALYZE PLAINTEXT CHARACTER STRINGS (235)
DETERMINE CONFIDENCE SCORE (240)
DETERMINE CERTAINTY LEVEL (245)
HIGH
HIGH | MEDIUM | LOW? (250)
LOW
MEDIUM
LABEL PLAINTEXT AS POTENTIAL PASSWORD (270)
VERIFY PASSWORD (275)
CONFIRMED PASSWORD? (280)
NO
YES
LABEL PLAINTEXT AS FALSE POSITIVE (255)
GO TO 225 (FIG. 3A)
LABEL PLAINTEXT AS PASSWORD (260)
UPDATE PARAMETRIC VALUES (265)
GO TO 220

Figure 3(a) & 3(b):

As seen in FIG. 2, the encryption unit 180 can be included in the remediation unit 190. The remediation unit 190 can be arranged to identify an appropriate remediation action based on the confidence score or certainty level. The remediation can include effectuating the encryption unit 180 to encrypt the computer resource (or CRA 20) containing the plaintext password. The remediation can consist of password verification, including initiating a password verification session with a computer resource asset at an operator location and receiving confirmation instructions or data from the operator via the computer resource asset. The remediation can include classifying the plaintext character string as a false positive. When the character string is determined to have a plaintext password, the remediation can consist of: annotating the character string with a label that indicates the character string consists of a confirmed password; interacting with the MT&T unit 170 to update the parametric values in the ML model; interacting with the MT&T unit 170 or DB 120E to update the training or testing datasets.

FIGS. 3A and 3B show a non-limiting embodiment of the plaintext password remediation process 200, according to the disclosure principles. Method 200 can be carried out by system 100 (shown in FIG. 2). Process 200 can begin by identifying the target CRA 20 (shown in FIG. 1) on the computer network 10 (Step 205) and searching one or more computer resources 22 on the CRA 20 (Step 210). All computer resources 22 that contain text can be searched by the plaintext search agent 160A (shown in FIG. 2).

To facilitate an understanding of the process 200, the following description is directed to a non-limiting example in which the computer resource 22 to be searched a configuration file. The CRA 20 is an end-user computer connected to the computer network, understanding that any other computer resources, including documents or files, or computer resource assets, can be targeted or searched for plaintext passwords.

The configuration file on the end-user computer can be searched by the plaintext search agent 160A (shown in FIG. 2) in any of several ways, as will be understood by those skilled in the art. For example, a list of known plaintext passwords can be retrieved from the DB 120E (shown in FIG. 2), and beginning with a first plaintext password entry on the list, each password on the list can be compared to all the text in the configuration file, to determine whether the configuration file includes a matching plaintext password.

Alternatively, the text in the configuration file can be filtered by the plaintext search agent 160A to remove or ignore all plaintext that is highly likely (for example, greater than 99.99%) not to contain plaintext passwords, such as, for example, common usage words like "the," "and," "from," "to," among others. The plaintext that remains after filtering can then be compared against each of the passwords on the list, or the passwords on the list can be compared against the plaintext in the configuration file.

After the plaintext search agent 160A finishes searching the text in the configuration file (Step 210), determining whether the configuration file contains a known password (Step 215). The plaintext search agent 160A can determine by comparing a list of known passwords against the text in the configuration file while searching the text; or the plaintext character analyzer 160B can determine after analyzing strings of characters in the text. If a known password is found in the configuration file, such as a password on the list (YES at Step 215), then a trigger signal can be sent to the remediation unit 190, and the configuration file can be encrypted (Step 220).

In an alternative embodiment, the plaintext password detector suite 160 can interact with the remediation unit 190 to detect and extract the plaintext password in the configuration file and replace it with the corresponding ciphertext token (Step 220). The ciphertext token can include encryption of the plaintext password by the encryption unit 180 (shown in FIG. 2) using an encryption algorithm. For example, the password-based key derivation function (for example, PBKDF1, PBKDF2 or scrypt), a password hashing function (for example, bcrypt), or any encoding technique that can protect and defend against attempts or attacks (for example, brute-force attacks) to decrypt or decode the plaintext password from the ciphertext token.

Upon completing encryption of the plaintext in the configuration file (Step 220), a determination can be made by the password detector suite 160, whether to search another computer resource on the same end-user computer or to search another computer on the computer network (Step 225). If it is determined that another computer resource or computer should be searched (YES at Step 225), then process 200 can be repeated with another computer resource on the same computer or a different end-user, otherwise the method 200 can end (NO at Step 225).

If, however, after finishing searching the text in the configuration file, a determination is made that the configuration file does not contain any known password (NO at Step 215). The text in the configuration file can be, optionally preprocessed (Step 230). The preprocessing can include eliminating unconcerned differences, such as, for example, non-password text such as common use words, or sizing and normalization of the plaintext character strings in the configuration file. The preprocessing can include parsing the plaintext in the configuration file into strings of plaintext characters. The preprocessing can be performed by either the plaintext search agent 160A or plaintext character analyzer 160B.

The plaintext character strings in the configuration file can be analyzed by the plaintext character analyzer 160B (Step 235) and a confidence score determined by the plaintext password predictor 160C for each string of plaintext characters file (Step 240). In analyzing the text in the configuration file, a string of characters, including each character's position and its relationship to all other characters in the string, can be analyzed to extract features that can be used to recognize and classify patterns. The plaintext characters can include characters from any known alphabet (for example, Arabic, Aramaic, Armenian, Brahmi, Cyrillic, Georgian, Greek, or Latin) or any known symbols (for example, braille, mathematics, or scientific). In a non-limiting example, the plaintext analyzer 160B (or ML model) can be trained to look for character strings that have (i) at least one uppercase letter, (ii) at least one lowercase letter, (iii) at least one number, (iv) at least one unique character, and (v) a total length of at least eight (8) characters, and apply a weight (for example, "0" or "1") for each factor (i) to (v) in predicting a likelihood that a character string contains a plaintext password.

FIG. 4 shows a non-limiting example of table 300 that includes the five factors (i) to (v) that can be accounted for when analyzing the strings of characters in the plaintext character analyzer 160B using the ML model to predict a plaintext password. In this example, table 300 can include a factor (vi) for the total number of characters in the string and whether the total number exceeds a maximum character length (for example, 24 characters).

While the example shown in FIG. 4 includes five factors with binary weights "0" or "1" applied for each factor that is satisfied, any number of factors, including more or less than five factors, can be utilized in the analysis. Similarly, each weight can have any range of values, including 0 to 100, which can be based on a predicted likelihood that the factor is satisfied.

Referring to the non-limiting example in table 300, when a character string having the value "js7ada#@8d" is found in the configuration file, the plaintext character analyzer 160B can parse and analyze. Each character, as well as the group, characters as a whole (Step 235) and determine that the string of characters meets factors (ii) to (v), but does not meet factor (i) in the table. That is, the analyzed character string (i) does not include any uppercase letters ("n") but does include (ii) at least one lowercase letter ("y"), (iii) at least one number ("y"), (iv) at least one unique character ("y"), and (v) at least 8 characters in total ("y"). Having learned previously that plaintext passwords included all five factors (i) to (v). The analysis can apply a weight of "0" form factor (i) and a value of "1" for each of the other elements (ii) to (v), and determine a confidence score of 4 out of 5 (or 80%) likelihood that the plaintext character string js7ada#@8d includes a plaintext password.

A certainty level can be determined byte password CL score unit 160D (shown in FIG. 2) for the character string having a predicted plaintext password based on the confidence score. It was thereby determining the level of certainty that that character string likely includes the plaintext password (Step 245). In the non-limiting example above, the password CL score unit 160D can be arranged to determine a certainty level having any one of three values: a high, medium, or low certainty level. As mentioned above, the high certainty level includes plaintext character strings for which the confidence score value is greater than or equal to 70% (up to 100%). The medium certainty level includes plaintext character strings. The confidence score value is greater than or equal to 30% but less than 70%, and the low level has plaintext character strings for which the confidence score value is less than 30%.

If the certainty level is determined to be low (LOW at Step 250), then that character string can be labeled by the password CL score unit 160D as a false positive (Step 255), and a determination made whether to end the process 200 (Step 225).

Suppose the certainty level is determined to be high (HIGH at Step 250). In that case, the plaintext character string can be labeled by the password CL score unit 160D as a plaintext password (Step 260). The password CL score unit 160D can interact with the MT&T unit 170 (shown in FIG. 2) to update the ML model (Step 265). The remediation unit 190 (shown in FIG. 2) to encrypt the configuration file, including the plaintext password (Step 220).

If the certainty level is determined to be medium (MEDIUM at Step 250), then the password CD score unit 160D can provisionally label the character string as a potential plaintext password (Step 270) and perform password verification of the character string (Step 275). The password verification can be performed automatically or through interaction with a human operator (not shown). In this regard, the password CL score unit 160D can interact with the remediation unit 190 (shown in FIG. 2) and trigger communication with the human operator via the IO interface 140 or driver suite 150. The remediation unit 190 can communicate the character string to the operator. For example, for display on a display device (not shown), and receive a confirmation entry from the operator, such as. For example, a confirmation that the character string includes a plaintext password, an assurance that the character string includes a false positive, or an annotation such as a label for the character string, which can be used by the MT&T unit 170 to update the ML model or training dataset stored in the DB 120E (shown in FIG. 2). The remediation unit 190 can be arranged to forward the confirmation entry to the password CL score unit 160D.

Suppose the received confirmation entry includes a confirmation that the character string includes a plaintext password (YES at Step 280). In that case, the password CL score unit 160D can convert the provisional label to a label that confirms the character string includes a plaintext password (Step 260). The ML model can be updated by the MT&T unit 170 with the label and plaintext character string (Step 265).

Suppose the received confirmation entry includes a confirmed false positive (NO at Step 280). In that case, the provisional label is converted by the password CL score unit 160D to a confirmed false positive (Step 255).

5. Conclusion

Passwords are the most popular authentication method, mainly because they are easy to implement, require no special hardware or software, and are familiar to users and developers. Unfortunately, most users store their sensitive information or credentials in plaintext that might be accessible to attackers. Since the information is not encrypted and stored or transferred in cleartext, attackers will easily read it. Keeping a plaintext password in a configuration file allows anyone to read the file access to the password-protected resource. The MLP will be able to detect a plaintext password in a character string by analyzing plaintext character strings for typical password complexity, such as, for example, including at least one uppercase letter, lowercase letter, number, unique character, and text length (for example, minimum of eight characters). The MLP will ensure data security and integrity by extracting the plain text secrets, which will be remediated based on the possibility percentage. It will also prevent unauthorized users from accessing confidential data, modifying and corrupting data, or causing system outages.

References

[1]. The Importance of periodic security assessment, nil - https://www2.nil.com/rs/511-FXO-494/images/NIL-Security-Assessments-services.pdf
[2]. Exposed Credentials, MS-ISAC - https://www.cisecurity.org/wp-content/uploads/2018/04/Security-Primer-Credential-Exposure.pdf
[3]. Password Plaintext Storage, OWASP - https://owasp.org/www-community/vulnerabilities/Password_Plaintext_Storage
[4]. Machine Learning What it is and why it matters, - sas, https://www.sas.com/en_us/insights/analytics/machine-learning.html
[5]. System for slowing password attacks, Clifford E. Kahn, Jeffrey C. Venable, Sr., Roger A. Chickering - https://patents.google.com/patent/US8312540B1
[6]. Password-protection module, Burton KaliskiMagnus Nystrom - https://patents.google.com/patent/US20060041759A1

Author's Biography

Nada Al-Noaimi is working as a cybersecurity specialist in the Department of Information Protection, Saudi Aramco, Saudi Arabia. She completed her Bachelor's degree in Information Technology at Prince Mohammed bin Fahad University. Her areas of interest are cybersecurity, specifically web application penetration testing.

Abdullah Al-Turaifi has 7 years of experience as a cybersecurity specialist working in the Department of Information Protection, Saudi Aramco, Saudi Arabia. He received his Bachelor's degree in Information Systems from King Saudi University, Saudi Arabia, and a Master of Science in Management Information Systems from Florida International University, USA. His areas of interest are application security, mobile security, and DevSecOps.

SireenBabateenis is working as an IT system analyst in the Department of Corporate Applications, Saudi Aramco, Saudi Arabia. She completed her Bachelor's degree in Cybersecurity and Forensic computing at the University of Portsmouth. Her areas of interest are Front End Web Development and Data Science.