MANAGING PASSWORDS USING HONEYWORD DETECTION SYSTEM

[1]Manisha Bhole , [2]Prof. Dr. Girish K. Patnaik
[1]P.G Student [2]H.O.D
[1]Computer Science and Engineering,
[1] SSBT'S COET,Bambhori,Jalgaon,India

Abstract : With an emphasis on Digital India and Government's encouragement for cashless transactions, it has become essential for organizations to maintain the secrecy of login credentials of their employees and clients. At the same time, it is also necessary to avoid any suspicious activities/attempts to steal such data by the unauthenticated user. There are many methods to achieve secure logins like OTP and Token generators. But these methods require additional devices to be carried by the authenticated user. Loss or change of the additional devices can obstruct the authenticated user from logging in. Organizations can increase the authenticate user's comfort while maintaining the secrecy by storing a bunch of decoy passwords or "honeywords" corresponding to the correct password in the hashed password database. For the purpose of 'password cracking' an unauthenticated user hacking the hashed password database would not be able to identify the decoy password and an attempted use of honeyword can set off an alarm. In the proposed work the Honeyword generation method i.e. chaffing-with- tweaking provide an enhanced password security as a solution to an open problem that also overcomes password-cracking problem.

***IndexTerms* – Authentication,password,security.**

I. INTRODUCTION

The term "honeywords" is a play on "honeypot", which in the information security really refers to creating fake servers and then learning how attackers attempt to exploit them in effect, using them to help detect more widespread intrusions inside a network. "Honeywords are a simple but clever idea". Seed password files with dummy entries that will trigger an alarm when used. That way a site can know when a hacker is trying to decrypt the password file. While there are many attacks against password protection systems, password cracking refers to the process of extracting passwords from data. The most straightforward attack is called the brute-force attack.Simply trying every possible combination of characters until you find a matching hash value. So may become very time consuming particularly with long passwords. The difficulty also increases when more characters are allowed in passwords. The numbers grow quite large as the length and complexity of the password increases and would seem to make password cracking impossible. While it's true that the length of time to brute-force passwords increases with complexity, there are several other techniques that crackers can use to expose these passwords.The RockYou attack, in particular, revealed millions of commonly used passwords and has become part of the standard dictionary used to crack passwords. Password leaks are becoming a common occurrence on the internet with several large-scale leaks happing every year. These leaks have revealed the poor practice many companies employ when storing their passwords. The widely available lists of common passwords, an expanding knowledge base on how user select passwords and advances in password cracking technologies have made basic hashes more vulnerable than ever.

Section 1.1 Describes an Overview of Honeywords. Motivation is described in Section 1.2. Section 1.3 Describes Contribution. Organization of the Report is described in Section 1.4. Finally, Summary of the chapter is given in the last section.

1.1 Overview of Honeywords

Basically, the simple but clever idea behind the study is insertion of false pass- words – called as honeywords – associated with each user's account. When an adversary gets the password list, she recovers many password candidates for each account and she cannot be sure about which word is genuine. Hence, cracked password files can be detected by system administrator if a login attempt is done with a honeyword by the adversary. Some honeyword generation methods were eplained as follows:

Chaffing-by-tweaking: The user password seeds the generator algorithm which tweaks selected char- acter positions of the real password to produce the honeywords. For instance, each character of a user password in predetermined positions is replaced by a randomly chosen character of the same type: digits are replaced by digits, letters by letters, and special characters by special characters. A number of positions to be tweaked denoted as t should depend on system policy. As an example t = 3 and tweaking last t characters may be a method for the generator algorithm Gen(k, t).

Chaffing-by-tweaking-digits: It is executed by tweaking the last t positions that contain digits. For example, by using the last technique for the password 42hungry and t = 2, the honeywords 12hungry and 58hungry may be generated.

Chaffing-with-a-password-model: In this approach, the generator algorithm takes the password from the user and relying on a probabilistic model of, [8] real passwords it produces the honeywords, [9]. The authors give the model of as an example of this

method named the modeling syntax. In this model, the password is split into character sets. For instance, mice3blind is decomposed as 4-letters + 1-digit + 5-letters L4 + D1 + L5 and replaced with the same composition like gold5rings.

Chaffing with "Tough Nuts": The system intentionally injects some special honeywords, named as tough nuts, such that inverting hash values of those words is computationally infeasible, e.g. fixed length random bit strings should be set as the hash value of a honeyword. An illustrative example of a tough nut is given, in [9] as '9,50PEe[KV.0?RIOtcL-:IJ"b+Wol¡*]!NWT/pb'. It is stated that the number and positions of tough nuts are selected randomly. By means of this, it is expected that the adversary cannot seize whole sweetword set and some sweetwords will be blank for her, thereby deterring the adversary to realize her attack.In [9] is discussed that in such a situation the adversary may pause before attempting a login with cracked passwords.

Hybrid Method: By using this technique, random password model will yield seeds for tweaking- digits to generate honeywords. For example, let the correct password be apple1903. Then the honeywords angel2562 and happy9137 should be produced as seeds to chaffing-by-tweaking- digits.

1.2 Motivation

Real passwords are often weak and easily guessed; either by sharing passwords, using names of loved ones, dictionary words, and brute force attacks. The first step is to study and identify websites that are actively targeted by account creation tools. Specifically, chose to target: Twitter, Facebook, Pinterest, LinkedIn, Wordpress, eBay, and Hotmail. All of these sites were well represented in the forums (i.e. many people were selling accounts from, and tools targeting these sites) and they are extremely popular with web users in general. To provide protection against online attacks generating honeywords against the password it may be the solution for these problems.

1.3 Contribution

The contribution in the proposed system is to make a situation for an unauthenticate user by making the protocol which increases the chance of login in dummy environment. Also restrict the unauthenticated user to login the system.

1.3.1 Problem Statement

To develop a system, a user must be authenticated after login the system. Whenever a user wants to login the system with username and password, the password is authenticated by an authenticator. Password authenticator checks the user's password stored in the database. If a password exists then the authenticator sends a reply to the user as successful. Otherwise, if the user password is similar to the original password stored in the database then the user is allowed into dummy environment. And if the password is not similar then the authenticator sends a reply to the user as unsuccessful.

1.3.2 Objectives

Objectives of the proposed work are as follows-• Increase the chances of unauthenticated user guessing a Honeyword.• Unauthenticated user access to dummy environment and thereby protecting the user account.

1.3.3 Proposed Solution

Password leakage problem can be avoided using honeywords. Creating Honeywords by randomly replacing the characters. Also creating a Dummy environment where files uploaded by an unauthenticated user will be deleted after the end of the session. And a user with the wrong password is given an error for the wrong password. A user with Honeyword is given access to Dummy environment. A user with the correct password is given access to an account.

1.4 Organization of the Report

The organization of the report is given as:-Chapter 1, titled Introduction, consists of sections like information about domain, motivation.Chapter 2, titled Literature Survey, consists the section background and related work.Chapter 3, titled Proposed solution, describes the proposed honeyword generation method. It also presents the flowcharts and algorithms used in the proposed methods.Chapter 4, titled Result and Discussion, describes implementation details such as experimental setup in terms of simulation environment, analytical results and discussion about experiment.chapter 5, titled Conclusion and Future Work, concludes and provides directions for further work.

1.5 Summary

In this chapter, the problem statement, proposed solution are discussed. In the next chapter, the Literature Survey is discussed.

II. LITERATURE SURVEY

On any computer system that controls resources for more than one user, the ability to authenticate different users is imperative. A password has long been the most common way users prove their identity to a computer. The attractiveness of password-based authentication lies not in its security, but in its simplicity, practicality, ease of use, and low cost. Businesses should seed their password databases with fake passwords and then monitor all login attempts for use of those credentials to detect if hackers have stolen stored user information. That's the thinking behind the "honeywords" concept first proposed in "Honeywords: Making Password-Cracking Detectable", a paper written by Ari Juels, chief scientist at security firm RSA, and MIT professor Ronald L. Rivest, who co-invented the RSA algorithm. The term "honeywords" is a play on "honeypot", which in the information security really refers to creating fake servers and then learning how attackers attempt to exploit them in effect, using them to help detect more widespread intrusions inside a network. "Honeywords are a simple but clever idea", said Bruce

Schneier. Seed password files with dummy entries that will trigger an alarm when used.

The chapter is organized as follows. Section 2.1 describes Background. Related Work is described in Section 2.2. Finally, Summary of the chapter is given in the last section.

2.1 Background

In password-based authentication, the identity of an individual is verified solely by his/her ability to present a previously agreed word. The results in a significant vulnerability. Security is compromised if an adversary learns a single word. The adversary who knows a user's password will be able to impersonate this user and to access the resources to which this user is entitled. An adversary may mount several types of attacks on password authentication systems.

Identification of three main categories, based on the target of the attacks,

1. Attacks on the system end: The type of attack is targeted at the passwords stored in the system. An example of this type of attack is password guessing attack

2. Attacks on the communication channel: These attacks target any communication channel through which passwords are transmitted. Definition of communication channel includes all devices, media and protocols which connect the user to the system which stores the password (or its hash). Examples are replay, eavesdropping, and man-in-the-middle attacks.

3. Attacks on the user end: These are directly targeted at the user. Examples are social engineering, shoulder surfing, dumpster diving, and phishing. For cracking purpose the attacker makes a guess as to the value of the original password. The attacker then hashes that guess using the appropriate password- hashing algorithm and compares the two hashes. If the two hashes match, the attacker has discovered the original password, or in the case of a poor password hashing algorithm, they at least have a password grant access. The two most commonly used methods to make these guesses are brute-force and dictionary attacks. With brute-force, the attacker attempts to try all possible password combinations. While this attack is guaranteed to recover the password if the attacker manages to brute-force the entire password space, it often is not feasible due to time and equipment constraints. If no salting is used, brute-force attacks can be dramatically improved through the use of pre- computation and powerful time-memory trade-off techniques, in [15] [17].

2.2 Related Work

Figure 2.1 shows the structure of literature survey. Password security form most users have a low awareness of their vulnerability and of the scope of damage that can be inflicted if their passwords are compromised. Password security can be achieved through measuring password strength, web password habit and honeyword generation technique.

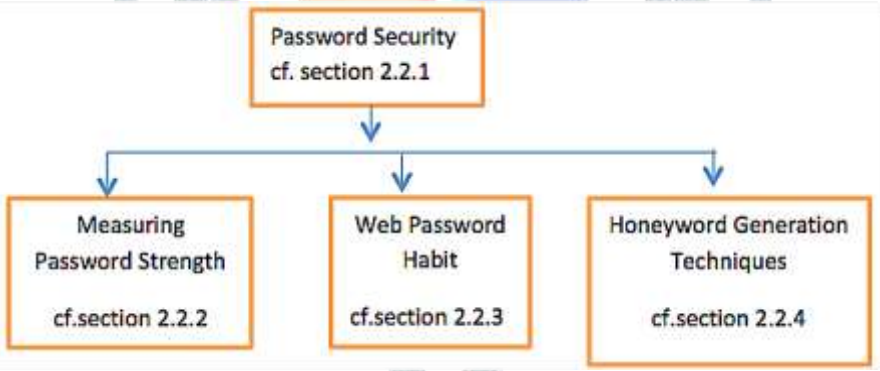

Figure 1: Structure of Literature Survey

2.2.1 Password Security

Notoatmodjo and Thomborson, in [7], presented user's perspective to their accounts and passwords. Authors described three main categories of attacks are, namely, attacks on the system end, attacks on the communication channel and attacks on the user end. By using passwords which they perceived to be more 'secure' on accounts that they considered important, participants demonstrated their awareness of the importance of using strong passwords to protect their valuable information.

Bojinov et al., in [1], presents kamouflage based password manager a new technique to prevent theft- resistant. The study states to use salts and slow hash functions to slow down a dictionary attack on the master password but unfortunately these methods do not prevent dictionary attacks. Authors states the main difficulties to overcome to make kamouflage work are, human-memorable passwords, related passwords, relation to master password and site restrictions. The authors have done with a survey that shows how users choose passwords. Authors have also described threat model, decoy set generation and fingerprinting. They ended with the conclusion stating kamouflage and fingerprinting technique provides security at high level.

Brown and Kelly, in [12], proposed that that the damaged caused by password leaks can be reduced by implementing few

good practices. In June 2012 many companies like LinkedIn, Yahoo etc... Were affected by password leaks which were publicly spread. Secure system should not have any loop holes that will allow intruders to get access to password files and should make sure that if the password hashes are been hacked it should not be easy to generate passwords from the hashes. Due to weak hashing mechanism these companies were highly affected is proved. In this paper the author has discussed about basics of password hashing and best practices that should be followed while password storage.

Kelley et al., in [9], proposed that the effects of password-composition policies on the guessability of pass- words. Seven different password-composition policies are used online to apply on a dataset of 1200 plaintext passwords.Described development approaches for calculating time consumed to guess each password they col- lected, also implemented guess-number calculator to evaluate the effectiveness of password-guessing attacks. Results also reveal important information about conducting guess-resistance analysis. Effective attacks on passwords created under complex or rare-in-practice composition policies require access to abundant, closely matched training data. Shannon entropy provides only a rough correlation with guess resistance and is unable to correctly predict quantitative differences in guessability among password sets.

2.2.2 Measuring Password Strength

Kelley et al., in [5], proposed that Seven different password-composition policies are used online to apply on a dataset of 1200 plaintext passwords. They have developed approaches for calculating time consumed to guess each password they collected. They have implemented guess-number calculator to evaluate the effectiveness of password-guessing attacks. Results also reveal important information about conducting guess- resistance analysis. Effective attacks on passwords created under complex or rare-in-practice composition policies require access to abundant, closely matched training data. Shannon entropy provides only a rough correlation with guess resistance and is unable to correctly predict quantitative differences in guessability among password sets. Password strength is a numerically expressed measure of how uncrackable a password is by considering the length and complexity of the password. Since password strength is measured by length and complexity, would it be safe to simply follow the generally recommended guidelines—use more than 8 characters and mix numbers, symbols, upper and lowercase letters. Altogether, there are 96 possible characters when choosing from A to Z in both upper and lowercase, 0-9, and all available keyboard symbols. A password with 8 characters could be any one of these 96, taken to the eighth power for the varying patterns. That means over 7,200 trillion password options! Not even a computer could easily handle cracking a password with that type of complexity.

Bonneau, in [3], proposed that the evaluation of large password data sets by collecting a massive password data set legitimately and analyzing it in a mathematically rigorous manner. In previous paper, Shannon entropy and guessing entropy not worked with any realistically sized sample, therefore, they developed partial guessing metrics including a new variant of guesswork parameterized by an attacker's desired success rate. In their study most troublesome is how little password distributions seem to vary with all populations of users.

2.2.3 Web Password Habit

Florencio and Herley, in [8], proposed that study of password used and password reused habits. They measured average number of passwords and average number of accounts each user has as well as measured number of times user enters password per day. They calculated this data and estimated password strength, password vary by site and number of times user forgotten password. In their findings, it showed users choose weak password; they measured exactly how weak. They measured number of distinct passwords used by a client vs. age of client in days also, number of sites per password vs. age of client in days. They also analyzed password strength. We are able to estimate the number of accounts that users maintain the number of passwords they type per day, and the percent of phishing victims in the overall population. The study involved half a million users over a three month period.

A client component on user's machines recorded a variety of password strength, usage and frequency metrics. Honeypot is one of the methods to identify occurrence of password database breach. The admin- istrator purposely creates user accounts to adversaries and detects a password disclosure if any one of the honeypot passwords get used in, [17], [18].

Idea has been modifed by Herley and Florencio, in [19], to protect online banking accounts from password brute-force attacks. According to the study, for each user incorrect login attempts with some passwords lead to honeypot accounts, i.e. malicious behaviour is recognized. For instance, there are 108 possibilities for a 8- digit password and let system links 10000 wrong password to honeypot accounts, so the adversary performing the brute-force attack 10000 times more likely to hit a honeypot account than the genuine account.

2.2.4 Honeyword Generation Technique

Bonneau, in [3], proposed that the evaluation of large password data sets by collecting a massive password data set legitimately and analyzing it in a mathematically rigorous manner. In previous paper, Shannon entropy and guessing entropy not worked with any realistically sized sample, therefore, they developed partial guessing metrics including a new variant of guesswork parameterized by an attacker's desired success rate. In their study most troublesome is how little password distributions seem to vary, with all populations of users.

Nagamalai and Dhinkaran, in [13], proposed that the characteristics of spam and technology used by spammers. They observed that spammers use software tools to send spam with attachment. To track and represent the characteristics of spam and spammers they setup a spam trap in their mail server. The paper is discussed in two types i.e. first type spam with attachment and second type is spam without attachment. They concluded, for spam without attachment, senders use non sophisticated methods

but for spam with attachment, senders use sophisticated software to spam end users.

Gencet al., in [11], proposed that hash passwords are used to improve security. For user authentication false passwords are added in hashed password file i.e. honeywords. They analyzed the honeyword system according to both functionality and the security perspective. They also elaborated how the system will respond to six password related attacks. Improvements for honeywords is described briefly i.e. number of honeywords, typo-safe honeyword generation and old passwords problem. Assumptions are illustrated to an active attack against honeyword system. They concluded that honeyword system is the powerful defense mechanism where an adversary steals the file of password hashes and inverts most or many of the hashes.

Zhao and Mannan, in [4], proposed that technology called Uvauth to hide authentication results from attackers to mitigate the risk of online password guessing. They propose the use of adapted distorted image as a computer-cipher/human-decipher channel to communicate short messages in human-machine interaction. The authors have discussed Uvauth and CAPTCHA for selfevidence of authentication that may make the scheme feasible. They have also elaborated possible attacks from attacker's perspective and some of them are limitations to current design. Limitations are they have not evaluated the server side load for generating and running a large number of fake sessions. They also have not tested how effectively users can detect implicit results from an authentication attempt, or whether messages via adapted distorted images can be used in practice.It can effectively deceive an attacker assuming fake sessions can be efficiently generated.

Juels and Rivest, in [2], proposed that honeywords technology to improve security level for authenticating fake users. The authors have also described briefly attacks on different scenarios, but have focused on stolen files of password hashes scenario. They have described various types of attacks on honeyword system that shows how it will manage and overcome it. The attacks are, namely, general password guessing, targeted password guessing, attacking the honeychecker, likelihood attack, DOS attack and multiple systems. The study shows to limit the impact of a DOS attack against chaffing-by-tweaking, one possible approach is to select a relatively small set of honeywords randomly from a larger class of possible sweetwords. Methods used . 1."Random pick" honeyword generation 2. Typo-safety 3. Managing old passwords 4. Storage optimizations. It inherit many of the well known drawbacks of passwords and something-you-know authentication more generally. Eventually, passwords should be supplemented with stronger and more convenient authentication methods, or give way to better authentication methods completely, as recently predicted by the media.

Erguler, in [14], proposed new honeyword generation algorithm in which honeywords are generated from the existing user passwords is proposed. It provides realistic honeywords. It shows better results with respect to flatness, DOS resistance and storage. New honeyword generation method 1. Chaffing-by-tweaking 2. Chaffing-with-a-password-model 3. Chaffing with "Tough Nuts" 4. Hybrid Method reduces storage cost of the honeyword scheme. It analyzed the security of the honeyword system and addressed a number of flaws that need to be handled before successful realization of the scheme. In this respect, we have pointed out that the strength of the honeyword system directly depends on the generation algorithm, i.e. flatness of the generator algorithm determines the chance of distinguishing the correct password out of respective sweetword.

Analyzed honeyword approaches and security of the system. Furthermore, point out that the key item for method is the generation algorithm of honeywords such that they were indistinguishable from the correct passwords. Therefore, propose a new approach in which the user's password is protected by creating honeywords. So the problems like hacking the authenticated user's password may be avoided.

2.3 Summary

In this chapter, background and related work about honeyword generation techniques are described. In the next chapter, Proposed Solution is presented.

III. PROPOSED SOLUTION

In recent past, Companies like Yahoo, Rockyou, LinkedIn, eHarmony, and Adobe has faced the problem of password leakage of millions of users. Disclosure of the password file is a serious security issue faced by the interested parties. To address the problem, the research proposes the use of Honeywords. Use of Honeywords addresses the issues in parts. First by creating an list of honeywords for a password created by the user and storing it in hash files. And second, by creating a dummy environment for situations where an unauthorized user tries to access the user files with honeywords instead on password.

Section 3.1 Describes a Proposed System Architecture. Proposed Algorithm For Honeyword Generation is described in Section 3.2. Section 3.3 Describes Proposed Algorithm For Honeychecker. Finally, Summary of the chapter is given in the last section.

3.1 Proposed System Architecture

To protect against the misuse of the user's real data conducted in a local file setting and password combination provides evidence that approach may provide unprecedented levels of user data security. Scrutinizing the honeyword system and highlights possible weak points. Also suggests the selection of the honeywords from existing user passwords in the system in order to provide realistic honeywords, a perfectly flat honeyword generation method where system stores all the passwords using honeywords. Figure 3.1 shows System architecutre for authentication of user. An adversary has obtained a password file F and cracked numerous user passwords. Tries to log in with any accounts in the list instead of compromising a specific account. The proposed protocol gives the detailed description about login conditions of authenticating the user, unauthenticated, and fake user.

User first submitting username and password to the system for authentication. For authentication puspose user must wait for sometime to get reply from authenticator. (Step-1 as shown in Figure 3.1)

Let, { U }= User, {Un}= Username, {X}= Password be tuples in the system architecture.

Authenticator forwords the username and password given by user to the honeychecker to verify a user.Authenticator must wait for sometime to get response from honeychecker. (Step-2 as shown in Fig- ure 3.1)

Honeychecker sends a request to the database to verify a user. To complete the task honeychecker access database. Database keeps record of all user's password and hash password. (Step-3 as shown in Figure 3.1)

Let, {H}=Honeychecker, D{X,Hp,Hn} be touples in the database

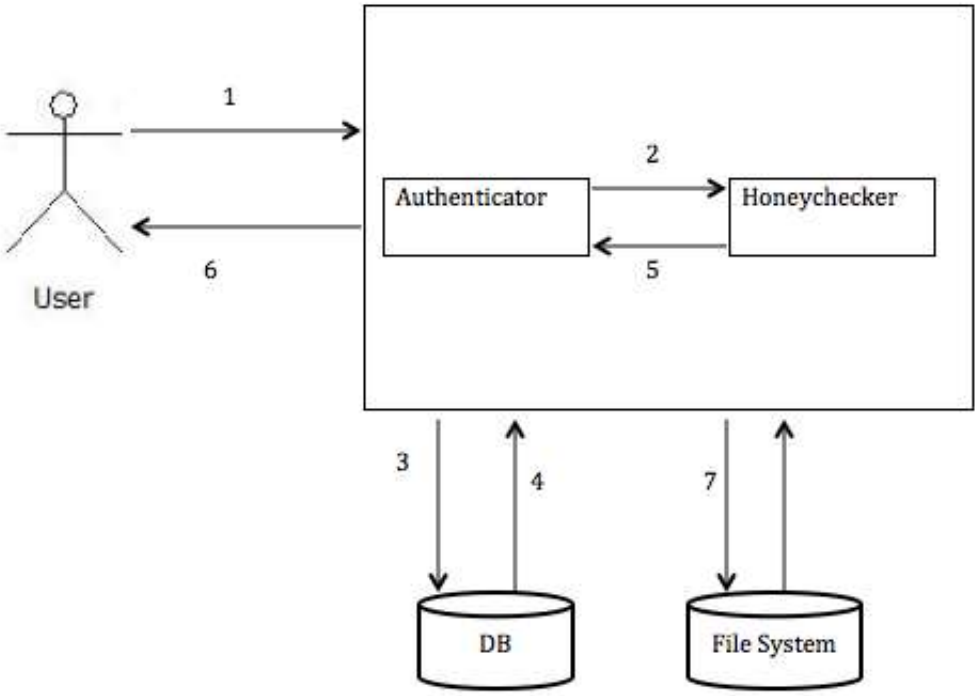

Figure 2: System Architecture for Authentication

Where,{Hn} = Honeywords,{Hp}= hash password Let, V is verification of a user Honeychecker verifies username and password from database. V=verify(X,Hp)

Honeychecker gets information from the database about the user.

If (X=Hp) then honeychecker sends a response to authenticator that user is authenticated user. Au- thenticator sends a response to the user to access account successfully. (Step-4 as shown in Figure 3.1)

If (X=Hn) then honeychecker sends a response to authenticator that user is an unauthenticated user. Authenticator sends a response to the user to access account. (Step-5 as shown in Figure 3.1)

If (X≠ Hp && X≠ Hn) then honeychecker sends a response to authenticator that user is fake user. Authenticator sends a response to the user to incorrect username and password. (Step-6 as shown in Figure 3.1)

Files may access by user is depends on the type of user. (Step-7, as shown in Figure 3.1)

Authenticated user can upload,download and view files any time user wants. Honeychecker tracks user's action based on the ip address. Also calculates cosine similarity between two session of login.

If (X=Hp) Then only authenticated user can access files. Let, {F} and {Dm} be a touple in the file system. Where, F be an actual files Dm be a Dummy File.

If (X=Hn) then a user may upload or download and view files in the very first login, After next login, all files should be removed from temp files. Files which are uploaded or download files by an unauthenticated user are in the corrupt format to make feel user like files damaged by virus attack.

3.2 Proposed Algorithm For Honeyword Generation

Algorithm 1 shows the procedure of creating a set of 'k' number of honeywords when a user creates an account. From model consider, user set a password 'x'. First, the length of the user password is calculated. Honeywords are created by randomizing the user password character with either alphabets or numbers or special characters. These honeywords are stored in sequence in the list. User password is also added to this list. Shuffle function is used to shuffle the sequence of these honeywords and password while returning the position of a password.

First user enters the username and password to the system. After entering the password by user, calculates length of password.

Let, U: User, a: length of password, X: Password by user Hence, a = length(U(X)) Selects random character from password. Whatever it is like alphabets, numbers or special character.

Based on that honeyword characters are being prepared. Let, C: character from password, P: password from database Hence, C=random(P); Generate Honeywords by collecting the character type from actual password. If password contains 4 alphabets, 3 numbers and 1 special caharcter, then honeywords which are created also contains same type of charatcters position. Alphabets are replaced by alphabets,numbers by numbers and special symbol by symbols.

Let, H: Honeyword, i: Position of password, C: character from password

Hence,

Hi = Cs

Where, s is the length of a password P. Fisrt length of password is decided (which is enter by user). After that honeywords are created.and 0<s<length(P)

Generates list of Honeywords. Let, L: Honeyword list, H: Honeyword Hence, L= Hk Where, 0<k<i shuffles honeyword list and actual password is also stored in same list then only login may successful otherwise fails. In database all 19 honeywords are generated sequntially from starting position, then actual password is also stored in honeyword list in the last position. For security purpose, list of honeywords gets shuffles to protect actual password.

Ls=shuffle(L)

(U(X) IN Ls)

Figure 3.3 shows honeyword generation flow. Take the real password of the user, then the real password is assigned a random position. Chaffing by tweaking logic is applied where the characters, which need to be replaced, are selected. Each character is assigned a randomly generated number between 33 to 126 ASCII characters. ASCII value of that randomly generated number is chosen from the standard ASCII and the original characters are replaced with ASCII characters. Steps are repeated for generating the honeywords for a real password.

Algorithm 1 Algorithm for Honeyword Generation

Require: U : User, H: Hash table, K: total number of passwords, X : Password by user, a : returns length of Password, i:Position of password, j=each character in passwords

1: Produce List(H) a = length(x)

2: for i= 1 to k-1, do

3: for j=1 to a, do honeyword(ij) = random(xj)

4: End for List(H:i)=honeyword(j) List (H:k) = add password x

5: End for

6: Shuffles List(H)

7: End Procedure.

3.3 Proposed Algorithm For Honeychecker

The system may incorporate an auxiliary secure server called the "honeychecker " to assist with the use of honeywords. Since assuming the computer system is vulnerable to having the file F of password hashes stolen. The honeychecker is thus a separate hardened computer system where such secret information can be stored. Honeychecker raises an alarm if a check fails. A system can communicate with the honeychecker when a login attempt is made on the computer system, or when a user changes the password. A communication is through dedicated lines and/or encrypted and authenticated. The honeychecker have extensive instrumentation to detect anomalies of various sorts. The honeychecker raising an alarm when an irregularity is detected. Honeychecker maintains a single database value for each user, the values are small integers in the range 1 to k, for some small integer parameter k (e.g. k = 20).

From Algorithm 2 Honeychecker starts action on any user authentication after generation of honeywords. Firstly honeywords list is generated, and from honeyword list honeychecker authenticate any user.

The user enters username and password and then request to the authenticator for verification. Authen- ticator forwards credentials of the user to honeychecker. Honeychecker checks that the first password is present in the honeyword list or not. Actual password must be present in the honeyword list. Honeychecker matches password with the hash password one by one. Whenever honeychecker found that password matches with the hash password honeychecker decide that user is authenticated the user. Accordingly, Send response to an authenticator, a user is a valid user.

If U(X) $\subseteq$ Ls && = H(X) then login successful for Authenticated user.Where, H(X): Hash passwordA user enters username and password and then request to the authenticator for verification. Authenticator forwards credentials of the user to honeychecker. Honeychecker checks that the first password is present in the honeyword list or not. Actual password must be present in the honeyword list. Honeychecker matches password with the hash password one by one. Honeychecker found that password matched with one of the honeyword in the same list. Accordingly, honeychecker decided that user must be an unauthenticated user.

If U(X) $\subseteq$ Ls &&$\vDash$ X then Login successful for unauthenticated user enters into dummy environment. Where, U(X) is user' password, Ls is honeyword list.A user enters username and password and then request to the authenticator for verification. Authenticator forwards credentials of the user to honeychecker. Honeychecker checks that the first password is present in the honeyword list or not. Actual password must be present in the honeyword list. Honeychecker matches password with the hash password one by one. Honeychecker found that password not matched with any honeyword or not with a hash password. Accordingly, honeychecker decided that user must be fake user.

If U(X) $\subseteq$ Ls && $\neq$ X then login Unsuccessful. Authenticator gives message to user, incorrect username and password. Where, U(X) is user's password, Ls is honeyword list. A user enters username and password and then request to the authenticator for verification. Authenticator forwards credentials of the user to honeychecker. Honeychecker checks that the first password is present in the honeyword list or not. Actual password must be present in the honeyword list. Honeychecker matches password with the hash password one by one. Honeychecker found that password not matched with any honeyword or not with a hash password. Accordingly, honeychecker decided that user must be fake user.

If U(X) $\not\subset$ Ls && $\neq$ X then login Unsuccessful. Authenticator gives message to user, incorrect username and password. Where, U(X) is user's password, Ls is honeyword list.

Algorithm 2 Algorithm for Honeychecker

Require: U: User, H: Hash table, K: total number of passwords, X : Password by user, a : returns length of Password, i: Position of password, j: each character in passwords X : Password by user a : length of passwordAssume U : User H : Honeyword L: Honeyword list k : Total number of passwords P: password from database c: character from password

1: Input user password U(X)
2: Generate length of password
3: Generate list of all Honeywords L.
4: Read password P from database,. Do
5: Select random character c from password P ,
6: Generate honeyword Hi
7: Generate list of all Honeywords L
8: Shuffle Honeyword list Ls while
9: all honeywords are generated.
10: If(U(X) IN Ls) then
11: Login success
12: Else
13: Login Fail

3.4 Summary

In this chapter, working and design of proposed approach and proposed system are described. In the next chapter, Results and Discussion are presented.

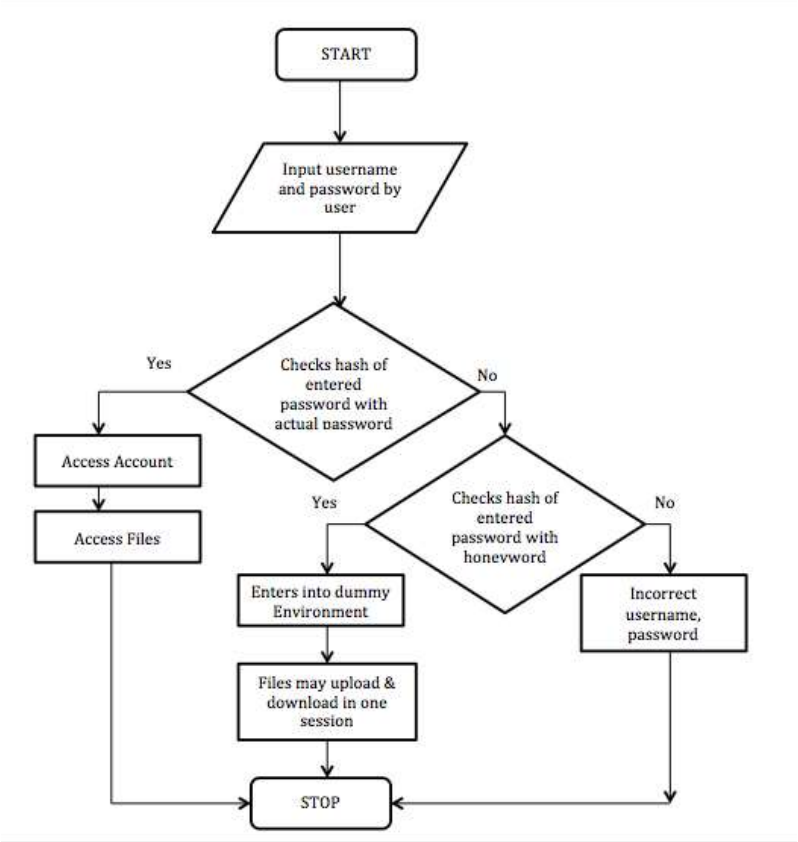

Figure 3.1 User's Authentication by Honeychecker

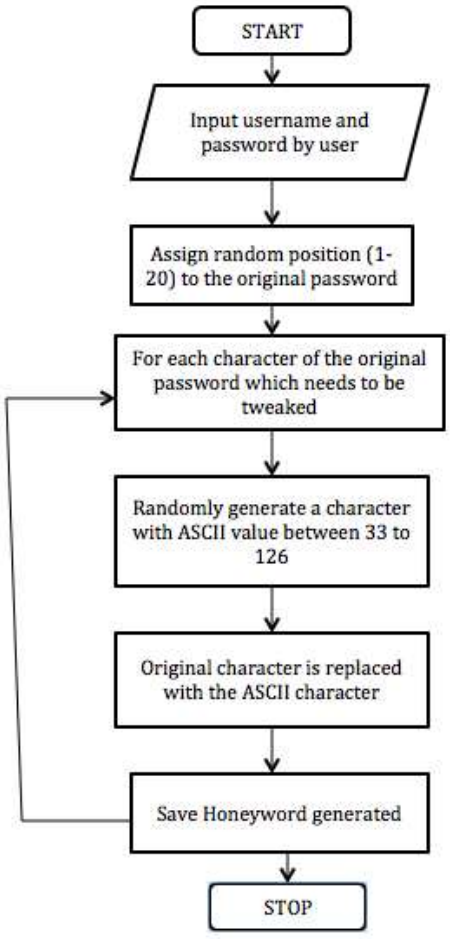

Figure 3.3 Flow of Honeyword Generation

IV. RESULTS

Result and discussion section is a primary part of research work. Evaluation of the proposed approach versus existing approach is carried out in the result and discussion section. Result section represents the experimental results of the proposed approach as well as the existing approach. Evaluation of all three approaches is carried out in the discussion section on the basis of obtained results. In some cases, various performance metrics are used to evaluate the system, in order to decide which one is the best.

Section 4.1 describes Implementation Details. The Experimental Setup is described in Section 4.2. Section 4.3 describes Analytical Results. Discussion of proposed work is described in Section 4.4. Finally, Summary is presented in the last section.

4.1 Implementation Details

The proposed system is implemented using JAVA, Apache Tomcat web server, MySQL Database. The variant of JAVA module JDK with version 7 is used for the implementation of the proposed solution. The Java API is grouped into libraries of related classes and interfaces. Tomcat implements several Java Server Pages (JSP) provides a pure Java web server environment for Java code to run. Tomcat is an application server from the Apache Software Foundation that executes Java servlets and renders web pages that include Java Server Page coding. My SQL is an open-source relational database management system.

4.2 Experimental Setup

The experimental setup is the component of research in which the experimenter analyzes the consequence of contribution on the existing system. Importance and primary aspect of the setup used in experiments are introduced in the experimental setup section.

4.2.1 Simulation Environment

Hardware Requirement- RAM-4GB DDR3, Hard Disk-1TB. Software Requirement- Front End-JSP, Server- Apache Tomcat, Database-MYSQL.

The Java environment is used for simulation of the algorithm on Intel i3 processor. Java is a high-level matrix or array language with control low statement function, data structure, input-output and object- oriented programming parameters. It allows both programmings in the small to create quick and dirty throw away programs and programming in the large to create complete large and difficult application programs. Set of tools and facilities makes it suitable to work with as the Java user or programmer.

4.2.2 Performance Metrics

Performance Metrics is used for the effectiveness to perform the function. Performance matric used in protocol are Password length, Possible number of Combination and Probability of correct password in 3 attempts. The possible number of Combination of a password is a different combination, which an algorithm will have to try to find the correct password and can be calculated by (4.1). The protocol of the proposed system increases the chance of login in dummy environment by the equation (4.2).

1. Possible Number of Combinations
   PNC = (L + N + S)n ….(4.1)

Where

PNC: Possible Number of Combination L: No of Alphabets N:No of Numerical S:No of Special characters

n:password length set by the user 2. Probability of Correct passwords in 3 attempts:

$$Pn = (1 - \frac{Hn}{PNC})(\frac{Hn}{PNC - 1}) \quad \text{…..(4.2)}$$

Where Pn: Probability of correct password Hn:No of Honeywords

4.3 Analytical Results

Passwords set by the users are between 8 to 16 characters, which include Letters in small or capital size, numbers from 0 to 9 and 11 special characters. The Table 4.1 shows the complexity created by the proposed system to increase the chances of guessing the Honeyword instead of the correct password. For the purpose, considered the data as shown below.

Table 4.1: Probability of Finding Password

Password Ln	No of combination	System probability	Honeyword probability	Increase in probability
8	6.45e15	1.55e-16	3.10e-15	5
9	5.30e17	1.89e-18	3.77e-17	5

10	4.30e19	2.33e-20	4.65e-19	5
11	3.45e21	2.90e-22	5.80e-21	5
12	2.75e23	3.64e-24	7.28e-23	5
13	2.17e25	4.60e-26	9.20e-25	5
14	1.71e27	5.85e-28	1.17e-26	5
15	1.34e29	7.48e-30	1.50e-28	5
16	1.04e31	9.61e-32	1.92e-30	5

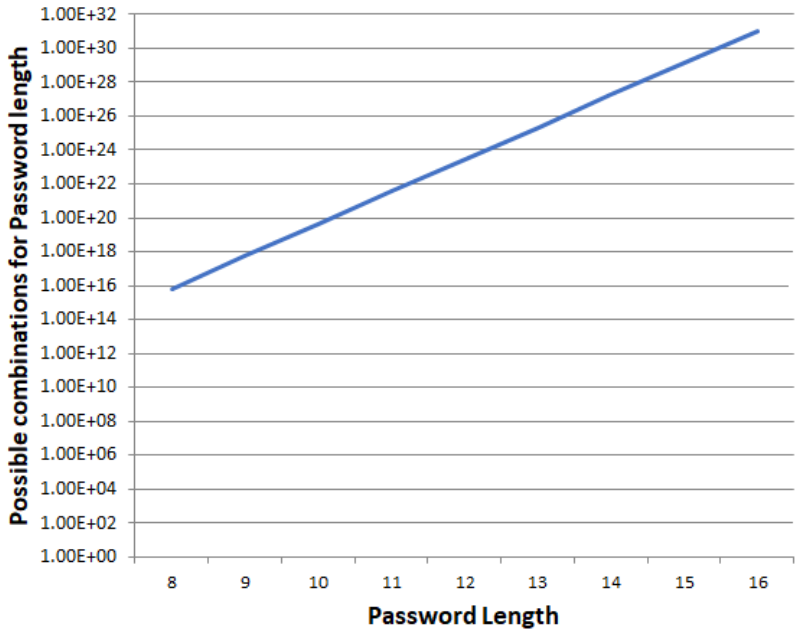

Figure 4.1: Password Length vs. Possible Combination for Password Length

Figure 4.1 shows the possible number of combinations against different password length. And Figure 4.2 shows its corresponding probability of finding correct password in 3 attempts in conventional single password encrypted in hash files. Figure 4.3 shows the net increase in chance of finding the honeyword is more by 5 percent and the difference between the probability improvement of finding a Honeyword.

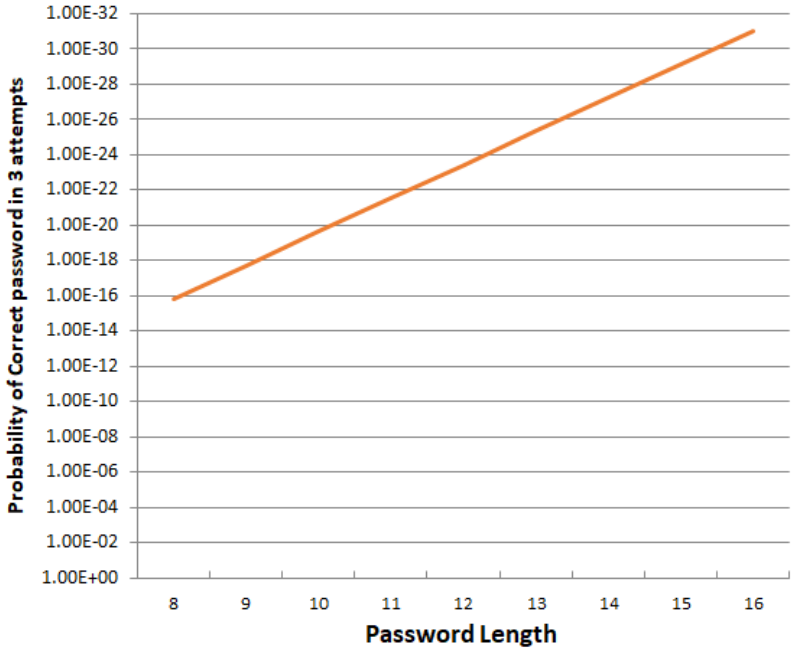

Figure 4.2: Password Length vs. Probability of Correct Password in 3 Attempts

4.4 Discussion

Results of the experiment show the effectiveness of the Honeywords in protecting the user account by giving them access to dummy environment to the hacker. As discussed in Experimental results, the intentions of Honeywords is to increase the chances of guessing the fake passwords instead of an actual user password. This is done so by creating the honeywords similar in character length of an actual password. Figure 4.1 shows the possible combinations that an unauthorized user can have for set password length. And Figure 4.2 shows the probability of finding the actual password in 3 attempts. Figure 4.3 compares the probability of conventional system and when honeywords are used for the same password length. By calculating the difference in the probability of a conventional password protection system and the probability of honeyword system, it can be said that there is increase 5% of guessing the honeywords and entering the dummy environment. Difference can be further increased by increasing the number of honeywords stored against the original password.

4.5 Summary

In this chapter, implementation details and experimental setup are described. In next chapter, Conclusion and Future Work are presented.

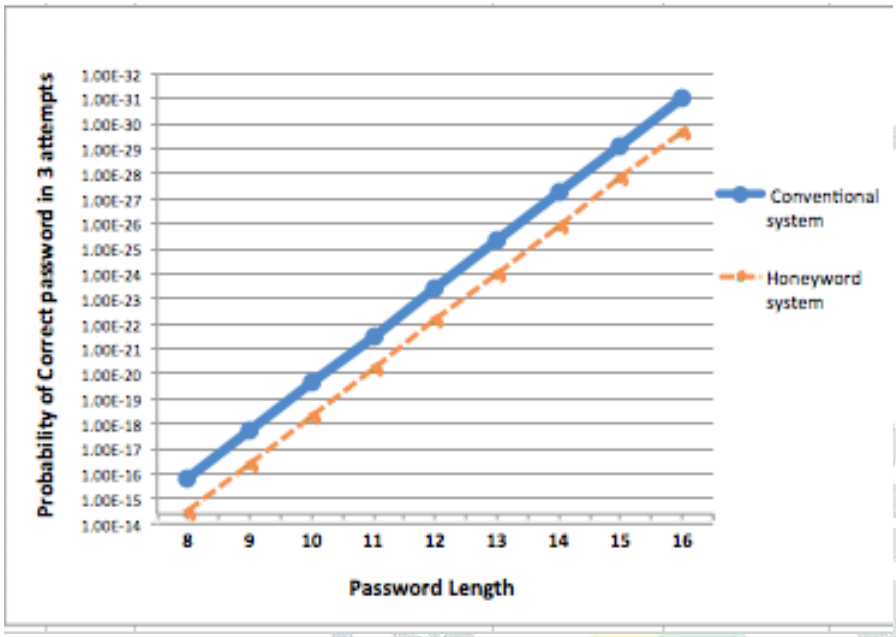

Figure 4.3: Probability Comparision of Single Password and Honeywords

V. Conclusion and Future Work

Honeywords becomes a various interesting challenging technique in security advantages over the password- based scheme. Proposed method takes less memory space and it can save storage overhead which is huge benefits. In the research made honeyword generation method 'chaffing-with-tweak' strong as compared to other research papers. By making each character replacement in the password, security of password must be increases.

The effectiveness of Honeywords depends on the algorithm. The risk of distinguishing user password from honeywords can be minimized by creating unpredictable honeywords. As an improvement and future scope of the project, Honeyword generation algorithm can be implemented with the selection of dictionary words in combination with numbers and special characters, equal in length of user password length.

References

[1] H. Bojinov, E. Bursztein, X. Boyen, and D. Boneh, Kamouflage: Loss-resistant Password Management,in Computer Security– ESORICS 2010. Springer, 2010, pp. 286–302.

[2] A. Juels and R. L. Rivest, Honeywords: Making Password-cracking Detectable,in Proceedings of the 2013 ACM SIGSAC Conference on Computer Communications Security, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 145–160. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516671

[3] J. Bonneau, The science of guessing: Analyzing an anonymized corpus of 70 million passwords,in Security and Privacy (SP), 2012 IEEE Symposium on. IEEE, 2012, pp. 538–552..

[4] L. Zhao and M. Mannan, Explicit Authentication Response Considered Harmful,in Proceedings of the 2013 Workshop on New Security Paradigms Workshop–NSPW '13. New York, NY, USA: ACM, 2013, pp. 77–86. [Online]. Available: http://doi.acm.org/10.1145/2535813.2535822

[5] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. Lopez, Guess again (and again and again): Measuring Password Strength by Simulating Password- cracking Algorithms,in Security and Privacy (SP), 2012 IEEE Symposium on. IEEE, 2012, pp. 523–537.

[6] J. Bonneau and S. Preibusch, The Password Thicket: Technical and Market Failures in Human Authen- tication on the Web, in WEIS, 2010.

[7] G. Notoatmodjo and C. Thomborson, Passwords and Perceptions,in Proceedings of the Seventh Aus- tralasian Conference on Information Security–AISC 2009. Australian Computer Society, Inc., 2009, pp. 71–78.

[8] D. Florencio and C. Herley, A Large-scale Study of Web Pass-word Habits,in Proceedings of the 16th international conference on World Wide Web. ACM Press, 2007, pp. 657–666.

[9] Kelley, Saranga Komanduri, Michelle L. Mazurek, Richard Shay, Timothy Vidas Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor and Julio Lopez, Guess again and again and again: Measuring password strength by simulating passwordcracking algorithms ,in IEEE Symposium on Security and Privacy, pp. 523-537, May- 2012.

[10] D. Malone and K. Maher, Investigating the Distribution of Password Choices,in Proceedings of the 21st International Conference on World Wide Web, ser. WWW '12. New York, NY, USA: ACM, 2012, pp. 301–310. [Online]. Available: http://doi.acm.org/10.1145/2187836.2187878

[11] Z. A. Genc, S. Kardas, and K. M. Sabir, Examination of a New Defense Mechanism: Honey- words,Cryptology ePrint Archive, Report 2013/696, 2013.

SSBT's College of Engineering and Technology, Bambhori, Jalgaon (MS) 23

[12] Brown and Kelly, The dangers of weak hashes," SANS Institute InfoSec Reading Room, Maryland US, pp. 1–22, Nov. 2013,[Online]. Available: http://www.sans.org/reading-room/ whitepapers/authentication/dangers-weak-hashes- 34412.

[13] D. Nagamalai, B. C. Dhinakaran, J. K. Lee,An In-depth Analysis of Spam and Spammers,International Journal of Security and its Applications,Vol. 2, No. 2, arXiv preprint arXiv:1012.1665, 2010.

[14] Imran ErgulerAchieving Flatness: Selecting the Honeywords from Existing User Passwords,IEEE Trans- actions on Dependable and Secure Computing, vol. 13, no. 2, pp. 284 - 295, February 2015.

[15] M.Hellman.Acryptanalytictime-memorytrade-off.IEEETransactionsonInformationTheory,Volume 26, Issue 4, pages 401-406, 1980.

[16] P. Oechslin. Making a Faster Cryptanalytic Time-MemoryTrade-Off. Proceedings of Advances in Cryp- tology (CRYPTO 2003), Lecture Notes in Computer Science, Volume 2729, pages 617-630, 2003. Springer.

[17] Cohen, F.: The use of deception techniques: Honeypots and decoys. Handbook of Information Security 3 (2006) 646–655.

[18] Almeshekah, M.H., Spafford, E.H., Atallah, M.J.: Improving security using decep- tion. Technical Report CERIAS Tech Report 2013-13, Center for Education and Research Information Assurance and Security, Purdue University (2013)

[19] C. Herley and D. Florencio, "Protecting financial institutions from brute-force attacks," in Proc. 23rd Int. Inform. Security Conf., 2008, pp. 681–685.