ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology(IJARTET)
Vol. 2, Issue 11, November 2015

Using Hashes With Patterns Techniques for Cracking Password

A.ARCHANA
M.Phil Research Scholar,
chana2591@gmail.com

Mrs.N.KOHILA M.Sc.,M.Phil.,MCA.,
Assistant Professor,
padmeshraj@gmail.com

Department of Computer Applications, Vivekanandha College of Arts and Science for Women (Autonomous).

***Abstract*— It is a common mistake of application developers to store user passwords within databases as plaintext or only as their unsalted hash values. Many real-life successful hacking attempts that enabled attackers to get unauthorized access to sensitive database entries including user passwords have been experi- enced in the past. Seizing password hashes, attackers perform brute-force, dictionary, or rainbow-table attacks to reveal plain- text passwords from their hashes. Dictionary attacks are very fast for cracking hashes but their success rate is not sufficient. In this paper, we propose a novel method for improving dictionary attacks. Our method exploits several password patterns that are commonly preferred by users when trying to choose a complex and strong password. In order to analyze and show success rates of our developed method, we performed cracking tests on real-life leaked password hashes using both a traditional dictionary and our pattern-based dictionary. We observed that our pattern-based method is superior for cracking password hashes.**

***Index Terms*—Password security, authentication, data security, dictionary attacks, hash cracking.**

I. MOTIVATION

AUTHENTICATION is one of the most important requirements for information security. There exist various methods for authentication based on what we know (e.g. pass-words, PINs), what we have (e.g. security hardware tokens) and who we are (e.g. Biometric fingerprints) [1]. Among the existing methods, password-based systems are easier to implement and therefore the most frequently used method for authentication. Being very critical for security, passwords are often targeted during cyber-attacks as well. An attacker that hacks a system and reveals user passwords stored within the database gets unauthorized access to accounts of all users. In the past many enterprise companies and organizations were victims of such attacks [2]–[6].

Attackers use frequently SQL injection vulnerabilities [7] that exist within applications in order to access database tables. They send arbitrary SQL queries to retrieve passwords and other sensitive data from tables and manipulate stored data, even by using automated tools such as sqlmap or Havij. Considering this fact, developers must never store passwords in plaintext within databases. Developers mostly know the fact that they should store hash values of passwords instead of plaintext. However, it is also a critical security weakness if the hash value of a password is calculated and stored without appending per-user unique salt value to the password before hashing [8]. In a classical scenario, a user chooses a password by a registration process. The hash value (md5, sha1, sha256 etc.) of the password is calculated on the backend- server and this calculated hash value is stored in the database. This implementation is very insecure too. Even though hash functions are one-way functions, attackers can perform brute- force, dictionary or rainbow-table attacks in order to reveal input values (i.e. plaintext password) from the given output values (i.e. hash value).

By brute-force attacks [9], the hash value of each possible input value is calculated and compared with the given hash value to crack. By dictionary attacks [10], large dictionary containing thousands or millions of possible passwords are utilized. Given a hash value to crack, an attacker calculates the hash value of each plaintext word from the dictionary line by line and compares the calculated hash values with the given hash value. If they are matched, the plaintext password is thus revealed. On the other hand, a very large set of pre-computed hash tables containing hash values and their corresponding plaintext values are used by rainbow-table attacks [11]. Given a hash value to crack, an attacker checks if the given hash value exists within the pre-computed lookup table. If it exists within the table, the plaintext password is found out.

If we compare brute-force, dictionary and rainbow-table attacks, they all have pros and cons. Brute-force attacks find out the plaintext definitely in the end but they are very time consuming. Dictionary attacks are fast but the success rate is not sufficient. Rainbow-table attacks are fast and successful at cracking but they require having a very big disk storage capacity. They are especially non-practical if a salt value is used for password hashes.

In this paper, we propose a new method for increasing success rates of dictionary attacks. For our method we analyzed leaked real-life user passwords and identified several patterns which are commonly chosen by many users to create a complex and strong password from a dictionary word. For example, a dot (“.”), an exclamation mark (“!”) or “123” are often appended at

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology(IJARTET)
Vol. 2, Issue 11, November 2015

the end of a dictionary word. Similarly, a dictionary word is repeated two times (e.g. kingking) or three times (e.g. kingkingking). We developed a software tool, namely pbp-generator (pattern based password generator), that implements our identified patterns and creates a new pattern-based large dictionary file from a given dictionary file. We generated a pattern-based dictionary file with ca. 2.3 billion passwords to crack password hashes belonging to different datasets which consist of real-life leaked password hashes.

Digital forensic investigators are involved with the analysis of crime cases. They often come across password protection during investigation. They need to crack passwords either in order to access a particular user account or to unlock encrypted or otherwise obfuscated digital evidence [12]. Our pattern-based method would help forensic investigators for more efficient password cracking.

It is important to note that security of hash functions is not within the scope of this paper. If a user chooses a weak password with a certain pattern, even a very secure hash function cannot prevent attackers from cracking password hashes. Patterns have no negative effect on computed hash values. In conclusion, the focus of this paper is the analysis of user-chosen plaintext passwords rather than the formal security model of hash functions.

This paper is organized as follows: Section II explains the details of how passwords and password patterns were analyzed. The identifi password patterns are explained in detail in Section III. Development of the software tool to generate pattern-based dictionary and perform hash-cracking tests with the generated pattern-based dictionary are explained in Section IV. Section V discusses the related work. Possible mitigation methods are given in Section VI. Section VII concludes the paper.

II. The Analysis

Rockyou.com web portal was the target of a very critical cyber-attack in December 2009 [3]. The hacker had found SQL injection vulnerability in the rockyou website and got access to its 32.6 million user passwords. Worse still, the passwords were stored as plaintext in the database. The leaked passwords without usernames were published in the Internet. In the past, security researchers did not have such a large real-life resource for password analysis. Therefore, the published 32.6 million real-life passwords have become a very valuable data for security experts and researchers.

A. Password Complexity Rules

It is always suggested that a secure password must not consist of only lowercase letters. Instead, it must contain lowercase and uppercase letters, digits and special symbol characters. A password fulfilling these complexity requirements would provide high entropy [13] and therefore should be more resistant against password guessing attacks. Today, enterprise companies and organizations define such password rules within their security policies and try to enforce their employees and customers to choose complex passwords.

TABLE I
The Top Ten List of Regular Expressions for Passwords With the Length Between 2 and 5

No.	Length=2 / Hit Count	Length=3 / Hit Count	Length=4 / Hit Count	Length=5 / Hit Count
	[a-z]{2} / 205	[a-z]{3} / 1664	[a-z]{4} / 8403	[a-z]{5} / 125731
		[0-9]{3} / 272	[0-9]{4} / 6359	[0-9]{5} / 44987
		[A-Z]{3} / 239	[A-Z]{4} / 889	[a-z]{3}[0-9]{2} / 18122
		[a-z]{2}[0-9]{1} / 82	[a-z]{3}[0-9]{1} / 603	[A-Z]{5} / 16006
		[A-Z]{1}[a-z]{2} / 63	[A-Z]{1}[a-z]{3} / 440	[a-z]{4}[0-9]{1} / 15791
		[a-z]{1}[0-9]{2} / 23	[a-z]{2}[0-9]{2} / 437	[A-Z]{1}[a-z]{4} / 5621
1		[a-z]{1}[0-9]{1}[a-z]{1} / 20	[a-z]{1}[0-9]{3} / 81	[a-z]{2}[0-9]{3} / 3092
2	[a-z]{1}[A-Za-z0-9]{1} / 3 [0-9]{2} / 49	[a-z]{1}[A-Za-z0-9]{1} [a-z]{1} / 19	[0-9]{1}[a-z]{3} / 64	[a-z]{1}[0-9]{4} / 2952
3	[A-Z]{2} / 45	[A-Z]{2}[0-9]{1} / 12		[0-9]{4}[a-z]{1} / 2272
4	[a-z]{1}[0-9]{1} / 13	[0-9]{1}[a-z]{2} / 9		[A-Z]{4}[0-9]{1} / 1951
5	[A-Z]{1}[a-z]{1} / 7			
6	[A-Za-z0-9]{2} / 5			
7	[A-Z]{1}[0-9]{1} / 5			
8	[a-z]{1}[A-Za-z0-9]{1} / 3		[0-9]{1}[a-z]{3} / 64	[a-z]{1}[0-9]{4} / 2952
9	[A-Za-z0-9]{1}[0-9]{1} / 2		[a-z]{2}[0-9]{1}[a-z]{1} / 53	[0-9]{4}[a-z]{1} / 2272
10	[0-9]{1}[a-z]{1} / 1	[0-9]{1}[a-z]{2} / 9	[a-z]{3}[A-Za-z0-9]{1} / 51	[A-Z]{4}[0-9]{1} / 1951

On the other hand, it is questionable if a password fulfi g the complexity rules including minimum length can be considered as a strong password. Let's take the following password "P4s5w0rd1." into consideration. This password has the length of ten characters and contains fi e lowercase letters, one uppercase letter, four digits and one special symbol. This password is considered and accepted in general as a strong password according to many password policies of enterprise companies and organizations. But we believe, this is an insecure password and can be easily cracked by using our pattern-based attack.

The password "P4s5w0rd1." contains three different common patterns. The first pattern is capitalization of the

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology(IJARTET)
Vol. 2, Issue 11, November 2015

first letter. The second pattern is replacing certain letters with numbers (a→4, o→0, s→5) and the third pattern is appending "1." to the password. Since people are bad
at remembering complicated passwords and have to use complex passwords due to password policies, they tend to create "strong" passwords by using such patterns. However, these common patterns jeopardize security of the passwords. If many passwords share the same patterns, they can be identified and then misused to guess passwords successfully with the help of automated tools.

B. Rockyou Pattern Analysis Based on Regular Expressions

Skull security [14] provides various leaked real-life password dictionaries to download. We utilized their special "rockyou" password list that includes additionally the total count for each unique password.

In the first step, we analyzed the rockyou passwords based on their regular expression representations. We created different Top 10 lists which consist of the most common regular expressions and their hit counts according to the different password lengths as shown in Table I, II and III. The Top 10 lists showed us some interesting facts. Most of the passwords are composed of appending numbers to letters. Therefore, we decided to continue with the analysis of dual and triple combinations of different character groups as explained in the following section. Another interesting fact is that the top one regular expression of passwords with the length of ten characters is ^[0-9]{10}$. This shows us that passwords belonging to this group consist of only numbers with the length of ten digits. We examined such passwords manually and concluded that these are mostly telephone numbers.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology(IJARTET)
Vol. 2, Issue 11, November 2015

TABLE II
THE TOP TEN LIST OF REGULAR EXPRESSIONS FOR PASSWORDS WITH THE LENGTH BETWEEN 6 AND 9

No.	Length=6 / Hit Count	Length=7 / Hit Count	Length=8 / Hit Count	Length=9 / Hit Count
1	[a-z]{6} / 601152	[a-z]{7} / 585013	[a-z]{8} / 687991	[a-z]{9} / 516830
2	[0-9]{6} / 390529	[0-9]{7} / 487429	[0-9]{8} / 428296	[0-9]{9} / 307532
3	[a-z]{4}[0-9]{2} / 215074	[a-z]{5}[0-9]{2} / 292306	[a-z]{6}[0-9{2} / 420318	[a-z]{7}[0-9]{2} / 273624
4	[a-z]{5}[0-9]{1} / 114732	[a-z]{6}[0-9]{1} / 193097	[a-z]{4}[0-9{4} / 235360	[a-z]{5}[0-9]{4} / 173559
5	[a-z]{2}[0-9]{4} / 98305	[a-z]{3}[0-9]{4} / 178304	[a-z]{7}[0-9{1} / 189847	[a-z]{8}[0-9]{1} / 160054
	[a-z]{3}[0-9]{3} / 98183	[a-z]{4}[0-9]{3} / 111218	[a-z]{5}[0-9{3} / 152400	[a-z]{6}[0-9]{3} / 132216
		[a-z]{1}[0-9]{6} /54883	[a-z{2}[0-9]{6} / 48541	[a-z]{3}[0-9]{6} / 44792
		[0-9]{6}[a-z]{1} / 41557	[A-Z]{8} / 39457	[A-Z]{9} / 27019
		[A-Z]{7} / 40592	[a-z{3}[0-9]{5} / 37622	[a-z]{4}[0-9]{5} / 22362
		[a-z]{2}[0-9]{5} / 32540	[A-Z]{6}[0-9{2} / 31373	[A-Z]{7}[0-9]{2} / 18482

C. ockyou Pattern Analysis Based on Dual and Triple Combinations

After analyzing the most common regular expressions representations, we analyzed the frequency of dual and triple combinations of different character groups (i.e. alpha, digitl and special symbol). In this analysis, [:alpha:] represents any alpha character between a to z and between A to Z. [:digit:] represents numbers between 0 and 9. [:symbol:] represents the following punctuation characters: . , " ' ? ! ; : # $ % & ( ) * + - / < > = @ [ ] ^ _ { } |.

By the dual combination analysis, the total numbers of [:alpha:]+[:digit:], [:alpha:]+[:symbol:] and [:digit:]+[:symbol:] combinations and their reverse order combinations were analyzed. This analysis showed us that circa 10 million rockyou passwords (30%) are in the form of [:alpha:] + [:digit:] combination, which means users mostly prefer appending a number to a dictionary word to create their passwords. Based on these results, we decided to examine [:alpha:]+[:digit:] combinations further to find more specific patterns. In the Table IV, the total counts of all dual combinations and their examples from the rockyou list are shown.

By the triple combination analysis, the total numbers of [:alpha:]+[:digit:]+[:symbol:], [:alpha:]+[:symbol:]+[:digit:] and [:digit:]+[:symbol:]+[:alpha:] combinations and their reverse order combinations were analyzed. Compared with the dual combinations, the triple combinations are not very much preferred by the rockyou users. The most frequently used triple combinations are [:alpha:]+[:symbol:]+[:digit:] with 0.57% and [:alpha:]+[:digit:]+[:symbol:] with 0.25%. Analyzing these combinations further we identified that digits and special symbols are together (e.g. "#1", "123.", "*1" etc.) appended to dictionary words to create passwords. The total counts of all triple combinations and their examples from the rockyou list are shown in Table V.

In addition to dual and triple combination analysis, we checked the frequencies of the punctuation characters. This analysis showed that certain symbols are more frequent than the others. The most frequently used punctuation character is point (.) with 0.7%. Underscore (_) has the second place with 0.58% and exclamation mark (!) has the third place with 0.55%. The total counts of each punctuation character in the password list are given in the Table VI. These frequencies were taken into consideration in our further analysis.

III. THE IDENTIFIED PATTERNS

The rockyou.com password list contains exactly 32,603,388 passwords. If the repeating passwords are

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology(IJARTET)
Vol. 2, Issue 11, November 2015

TABLE V
TRIPLE COMBINATION OF CHARACTER GROUPS WITH EXAMPLES

Combination	Total Count	Example Passwords
[:alpha:]+ [:digit:]+ [:symbol:]	82,151 (0.25%)	teenager1@, abc123., karl143., windowsxp1!, kelvin258/, jessie18;, pretti7*, jordans07., JUNE24,, briana20.
[:alpha:]+ [:symbol:]+ [:digit:]	185,610 (0.57%)	kaitlyn.1, poopp<3, t=48697123, franco_1, dude!2, chris#6, tommy.2359, iloveyou*1, Summer#5, watru^2
[:digit:]+ [:alpha:]+ [:symbol:]	13,298 (0.04%)	1hawaiian!, 1wish!, 072305AJ$, 1TIKA!!, 4evergreen!!, 123abc., 1love!, 707sucks!, 123loveme!, 1fighter/, 50cent., 1andonly.
[:digit:]+ [:symbol:]+ [:alpha:]	18,218 (0.06%)	11!!JesusS, 6.five, 555-oup, 7-boss, 1!iloveyou, 1*princess, 305-boy, 123!qaz, 100%jumper, 1986@Jessica, 15-red
[:symbol:]+ [:alpha:]+ [:digit:]	9,940 (0.03%)	.disney2, @$$baba82, *k123456, $hortii88, *supergirl12, *ILOVEYA7, *june7, $iloveu40, !batman76, @love2
[:symbol:]+ [:digit:]+ [:alpha:]	12,592 (0.04%)	#1CHRIZ, #1kingsfan, <3ilovemanuel, !11Mom, *789ab, #1hawaiian, #1carlos, #1lover, #1lady

TABLE VI
TOTAL COUNTS OF PUNCTUATION CHARACTERS WITHIN ROCKYOU PASSWORDS

.	**226,980 (0.70%)**	,	27,722 (0.09%)	"	3,172 (0.01%)	'	16,097 (0.05%)
!	**179,666 (0.55%)**	;	14,378 (0.044%)	:	7,239 (0.022%)	#	60,016 (0.18%)
%	11,282 (0.03%)	&	28,553 (0.088%)	(	16,557 (0.05%)	)	18,349 (0.056%)
+	24,000 (0.073%)	-	126,908 (0.39%)	/	37,836 (0.12%)	<	11,856 (0.036%)
=	18,741 (0.057%)	@	10,4336 (0.32%)	[	7,722 (0.02%)	]	10,731 (0.033%)
^	5,863 (0.018%)	_	**187,603 (0.58%)**	{	1,056 (0.003%)	}	933 (0.003%)
~	5,823 (0.018%)	$	31,501 (0.1%)	>	2,755 (0.008%)	|	506 (0.002%)
?	24,744 (0.08%)	*	95,400 (0.3%)				

eliminated, there are exactly 14,344,399 unique passwords. We examined thousands of passwords for possible patterns during our analysis of regular expressions and dual/triple combinations. Furthermore, we checked manually around 500 thousand passwords out of 14.4 million unique passwords to find additional patterns. The rockyou list of Skullsecurity was sorted according to the most frequently used password order. Therefore, the main password patterns exist already within our analyzed 500 thousand passwords group.

As a result, we identified several patterns which belong mainly to ten categories. These are *Appending, Prefixing, Inserting, Repeating, Sequencing, Replacing, Reversing, Capitalizing, Special-format* and *Mixed Patterns.*

A. Appending Pattern

The dual combination analysis showed that ca. 30% of all

TABLE VII
APPENDING PATTERN EXAMPLES

Pattern Example	Total Counts	Password Examples (Total Count)
Appending [0-9]	2,802,484	password1 (11,112), princess1 (5,187), angel1 (4,320)
Appending 123	353,400	abc123 (16,648), love123 (2,939), red123 (2,089)
Appending 1234	59,892	abcd1234 (1,322), abc1234 (518), love1234 (511)
Appending 101	55,521	love101 (820), zoey101 (700), sexy101 (616)
Appending dot	69,300	password. (484), iloveyou. (467), fuckyou. (135)
Appending !	104,273	iloveyou! (1,358), password! (701), rockyou! (485)

TABLE VIII
PREFIXING PATTERN EXAMPLES

Pattern Example	Total Counts	Password Example (Total Count)
Prefixing [0-9]	219,035	1password (874), 1bitch (752), 1lover (550)
Prefixing 123	26,306	123abc (4,115), 123qwe (1,614), 123asd (315)
Prefixing #1	8,617	#1bitch (333), #1pimp (121), #1hottie (119)
Prefixing !	2,967	!password (11), !basketball (7), !iloveyou (7)
Prefixing dot	1,113	.password (11), .adgjm (10), .iloveyou (4)

TABLE IX
INSERTING PATTERN EXAMPLES

Pattern Example	Password Example (Total Count)
Inserting [0-9]	love4ever (1,276), my3kids (579), my2kids (450)
Inserting 123	abc123abc (96), abc123def (27), abc123xyz (25)
Inserting dot	c.ranoldo (272), dr.pepper (232), man.utd (42)
Inserting #1	my#1love (12), my#1baby (9), my#1angel (5)

rockyou passwords are in the form of [:alpha:] + [:digit:] combination. Analyzing this special dual combination further, we identified many password examples of appending pattern, where a certain digit or punctuation character (or digit/ character groups) is added at the end of a dictionary word.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology(IJARTET)
Vol. 2, Issue 11, November 2015

Table VII gives the total counts of passwords belonging to this pattern and some password examples.

Among all patterns we identified, this pattern is the most frequent one. For example, about 2.8 million passwords are combinations of alpha characters with one or more digits.

B. Prefixing Pattern

The dual combination analysis showed that there are around 900 thousand passwords having the form of [:digit:]+[:alpha:] combination. Analyzing this special dual combination further,
we identified many password examples of prefixing pattern by which a certain digit and/or punctuation character (or digit/ character groups) is added at the beginning of a dictionary word. Table VIII gives the total counts of passwords belonging to this type and some password examples.

C. Inserting Pattern

In addition to appending and prefixing patterns, we identified many password examples of inserting pattern by which a certain digit and/or punctuation character (or digit/ character groups) is inserted into a dictionary word. Table IX gives some password examples of this pattern. Since distin- guishing inserting patterns from replacing patterns requires

TABLE X

REPEATING PATTERN EXAMPLES

Pattern Example	Password Example (Total Count)
Repeating number "N" for N times	1 (29), 22 (6), 333 (19), 4444 (58), 55555 (4,258), 666666 (7,419), 7777777 (4,589), 88888888 (2,493), 999999999 (1,952)
Repeating number groups	123123 (9,516), 303030 (678), 292929 (678), 420420 (669), 007007 (652), 789789 (634), 123456123456 (188)
Repeating [0-9] numbers	111111 (13,272), 11111 (5,003), 11111111 (1,512), 222222 (4,486), 22222 (1,116), 333333 (2,704)
Repeating birth years	19871987 (433) , 19891989 (429), 19921992 (416), 19861986 (399), 19931993 (380)
Repeating words	lovelove (2,742) , catcat (564), kisskiss (826), oneone (64), twotwo (32), passwordpassword (43), usausa (58), blablabla (481)
Repeating letter groups	abcabc (118), abcabcabc (20), ABCABC (6), defdef (8), defdefdef (2)
Repeating [a-z]	aaaaaa (2,685), aaaaa (1,252), bbbbbb (445), bbbbb (192), bbbbbbbb (77), bbbbbbb (66)
Repeating symbols	..... (343), ... (74), ... (41). ———— (35)

manual analysis (e.g. passw0rd vs. pass4word), the total counts for pattern examples are not given in the table.

D. Repeating Pattern

In addition to dual and triple combination analysis, we examined passwords which contain only alpha, digit or punctuation characters. This analysis showed that 44% of all passwords consist of only alpha characters, 16% contain only numbers and 0.015% contain only punctuation characters.

Since it is a known fact that users prefer choosing passwords without numbers and special symbols, 44% was an expected result for only-letter passwords. On the other hand, 16% seemed very unusual for passwords containing only digits.

Analyzing the passwords in this group further, we found out that some users tend to choose certain number combinations (e.g. 29, 1980, etc.) and repeat them to create a password. For example, a birth year is chosen and repeated (e.g. 19791979). We also realized that not only numbers, but words and punctuation characters are repeated as well to create passwords. As an example, a dictionary word is repeated two or three times (e.g. kingkingking). Table X shows some examples of repeating pattern.

E. Sequencing Pattern

In the analysis we identified the sequencing pattern by which sequences of keyboard layouts, alphabet letters, digits or their combinations are used to create passwords (e.g. qwerty, 123abcd, abcdqwer, etc.).

The most frequent keyboard sequence is “qwerty” with 13,456 passwords. The most frequent letter sequence is “abcdef” with 2,733 passwords. The most frequent digit sequence is “123456” with 290,729 passwords. This is the number one password in the Top 10 list.

Table XI shows examples of the sequencing pattern for keyboard layouts, alphabet letters, digit sequences and their combinations.

TABLE XI

SEQUENCING PATTERN EXAMPLES

Pattern Example	Password Example (Total Count)
Keyboard Sequences	qwerty (13,456), qwertyuiop (2,871), qwert (1,375), azsxdcfv (63), asdfgh (2,908), asdfghjkl (2,537), asdfg (1,190), zxcvbnm (3,521), zxcvbn (1,552)
Keyboard Sequences mixed with Digit Sequences	1q2w3e4r (1,205), 1Q2W3E4R (35,1234), qwer (545), asdf1234 (474)
Alphabet Letter Sequences	abcdef (2,733), abcdefg (1,856), abcde (955), abcdefgh (666), zyxwvu (8)
Alphabet Letter Sequences mixed with Digit Sequences	abcd1234 (1,322), a1b2c3 (688), 1234abcd (463), a1b2c3d4 (361), a1b2c3d4e5 (140)
Digit Sequences	123456 (290,729), 12345 (79,076), 123456789 (76,789), 1234567 (21,725)

TABLE XII

REPLACING PATTERN EXAMPLES

Replaced Letter	Replaced with	Password Examples
a	4	d4niel, c4r0lin4, dr4gon, pl4yboy
a	@	p@ssword, t@ylor, f@mily, b@bygirl, c@rlos, wh@tever, p@trick, eleph@nt, di@mond
b	6	straw6erry, sexy6lue, septem6er, remem6er
e	3	monk3y, socc3r, spong3bob, princ3ss, ilov3you
g	6	soccer6irl, hun6ry, ran6ers
g	9	an9els, en9ine, dan9er, babi9irl, magic9irl
i	1	1loveyou, , mel1ssa, stup1d, denn1s, w1lliams, pr1ncess, jess1ca, v1ctor, sn1ckers, sw1mm1ng
i	!	pr!ncess, m!chelle, just!n, sunsh!ne, pr!nce, jess!ca, babyg!rl, w!lliam, tw!ster
i	|	M|ChElLe, m|r@c|eS, sl|ther
l	1	P1ayer, ash1ey, a11star, isabe11a, ye11ow, wi11iam
l	|	love|y, my|ove, actual|y, m|r@c|eS, josh|and
o	0	il0veyou, ge0rge, m0vie, br0ken, passw0rd, c0llege, br0ther, n0thing, t0psecret, m0nkey
s	5	pas5word, du5tin,ju5tin, east5ide,augu5t, it5easy, ec1ip5e, chee5e
s	$	$prite, be$tfriend, jutin, twohort, $pecial, $ummer, $upersonic, $tevenrules

F. Replacing Pattern

In the analysis of inserting pattern we realized that certain letters are replaced with a number or a symbol. As an example, the letter “o” is replaced with the number zero (e.g. password → passw0rd). Similarly, the letter “s” is replaced with “$” or “fi e (5)” (e.g. sport → $port, august → augu5t). We examined this pattern further in order to identify more replacements. Table XII lists the identified replacement possibilities and their example passwords from the rockyou list.

G. Capitalizing Pattern

By this pattern some lowercase letters of a dictionary word are exchanged with their uppercase equivalents. As examples, the word “password” can be converted into “Password”, “passWord” or “passworD”. Providing this, such passwords become compliant with password policies which require passwords to contain at least one uppercase letter. More passwords examples of this pattern from the rockyou list are given in Table XIII.

TABLE XIII

CAPITALIZING PATTERN EXAMPLES

Pattern Example	Password Examples (Total Count)
Capitalization of 1st letter	Password (806), Princess (769), Jessica (471), Michael (410), Nicole (373), Daniel (368), Liverpool (350), Danielle (347), Michelle (345)
Capitalization of the last letter	rockU (15), passworD (7), whoamI (4), princesS (4)
Capitalization of the 2nd letter	iTunes (4), pAssword (3), iLoveyou (3), kEvin (2)
Capitalization of the 2nd word	RockYou (658), HarryPotter (94), iloveJesus (30), passWord (7)

TABLE XIV

SPECIAL-FORMAT PATTERN EXAMPLES

Pattern Example	Password Examples
Victim web page	rockyou, ROCKYOU, RockYou, Rockyou, rockyo, rockyouu
www sites	www.com, www.hotmail.com, www.yahoo.com, www.rockyou.com, www.google.com, www.hi5.com
E-mail addresses	wildcat_41@hotmail.com, sam@hotmail.com, ymeltzer@gmail.com, sfink1@gmail.com
Dates	4/30/04, 4/19/1992 , 29/12/91 , 19/03/1988
Sportsman player names with shirt numbers	ronaldo7, Ronaldo7, cristianoronaldo7, messi10, leonelmessi10, zidane10, jordan23, JORDAN23, Jordan23
Birth month with days or years	january14, january23, january19, february14, february23, february22, jan2005, jan2007, dec26, december2003
Telephone numbers	09001728888, 08123456789, 01478520369

H. Reversing Pattern

By this pattern dictionary word letters are put in a reverse order. As an example, the word "password" is converted into "drowssap". Some examples of this pattern from the rockyou list are as follows: drowssap, uoykcor, fedcba, elgoog, uoyevoli, ssecnirp, yraunaj, ylevol.

I. Special-Format Pattern

The last identifi pattern is special-format pattern. This pattern group contains passwords having special formats like dates in various forms (e.g. dd/mm/yy, mm/dd/yy, dd/mm/yyyy etc.), combinations of a birth month with a day or year in different forms (e.g. january15, jan15, jan2007 etc.) and combinations of a sportsman player name with his/her shirt number etc. Some examples of the special-format patterns with password examples from the rockyou list are given in Table XIV.

J. Mixed Patterns

This pattern represents mixing of two or more pattern types. Capitalization with reversing (e.g. droW) and capitalization with insertion (e.g. Wo2rd) are two examples. Table XV gives more examples of this pattern from the rockyou list.

IV. CRACKING TESTS WITH THE IDENTIFIED PATTERNS

After identifying several patterns, we proceeded with the benchmark of the identified patterns. We checked if they can improve efficiency of dictionary attacks by cracking more real-life passwords hashes that were leaked from different web portals in the past.

TABLE XV

EXAMPLES OF MIXED PATTERNS

Pattern Example	Password Examples
Appending with prefixing	123password123, 1password1, 1rockyou1, 12rockyou12
Capitalization with insertion	Love4ever, Jesus4life, My2girls, Jesus4me, My3sons
Reversing with replacing	dr0wssap, uoykc0r, 0uyev0li
Capitalization with reversing	drowssaP

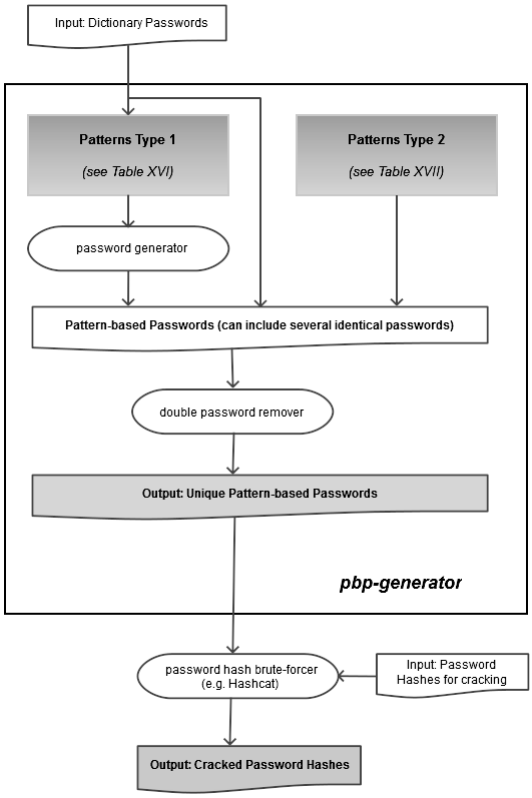

Fig. 1. Architecture of pbp-generator.

A. pbp-Generator (Pattern Based Password Generator)

We developed a software tool namely *pbp-generator* for benchmarking. As shown in Figure 1 pbp-generator gets a dictionary file as input, creates several variations of each dictionary word from the given input file based on Type 1 identified patterns (see Table XVI) and adds them to the output fi which represents the generated pattern-based dictionary file. Additionally, pbp-generator adds many other passwords from Type 2 identified patterns (see Table XVII) into the output file. The Type 2 patterns are not applied on input file, but they are used to create certain passwords (e.g. month name with year, special keyboard sequences etc.) to be added directly into the output dictionary file. The passwords from the given dictionary file are explicitly included within the output dictionary fi because this enables us to distinguish if a given hash can be cracked only with the pattern-based dictionary but not with the given input dictionary file. Before the output file is finalized, double passwords are removed and therefore the final file includes

TABLE XVI
TYPE 1 IMPLEMENTED PATTERNS (PATTERNS THAT ARE APPLIED TO EACH WORD IN THE DICTIONARY FILE)

Pattern Type	Implementation Details
Appending and Prefixing	0,1,2,3,4,5,6,7,8,9,10,123,101,010,132, 111, 1111, 11111, 222, 2222, 22222, 333, 3333, 33333, 444, 4444, 44444, 555, 5555, 55555, 666, 6666, 66666, 777, 7777,77777, 888, 8888, 88888, 999, 9999, 99999, #1 1. 1! . ! , ; & | ( ) * + - < > = [ ] ^ { } ? $ % : # / @ _
Repeating	2-times, 3-times, 4-times, 5-times (e.g. catcatcat)
Replacing	All at once (e.g. p445w0rd) replacement and single character at one time (e.g. passw0rd) replacement based on Table XII.
Reversing	Each word in the dictionary file is reversed.
Capitalizing	The first and the last letter of each dictionary word are capitalized.

TABLE XVII
TYPE 2 IMPLEMENTED PATTERNS (PATTERNS THAT ARE DIRECTLY ADDED TO THE OUTPUT DICTIONARY FILE.)

Pattern Type	Implementation Details
Combination of month name and day	Months between January and December are combined with a number between 1 and 31 (e.g. january26, may30)
Combination of sport player name and his/her shirt numbers	zidane10, ronaldo7, ramos4, adebayor6, kaka8, benzema9, alonso14, canales16, higuain20, dimaria22, ozil23, alves2, pique3, puyol5, xavi6, davidvilla7, iniesta8, messi10, forlan7, aguero10, simao20, johnson2, agger5, suarez7 etc.
Adding special sequences	1q2w3e4r, qawsedrf, qaswedfr, zaxscdvf, zasxcdfv, a1b2c3, a1b2c3d4, 1a2b3c, 1a2b3c4d, 1234qwer, 1234asdf, 1234zxcv, qwer1234, asdf1234, zxcv1234, 5678abcd, abcd5678, 5a6b7c8d, 5a6b7c, a5b6c7, a5b6c7d8
Adding victim web sites	Mayhem, mayhem, stratfor, Stratfor, Rootkit.com, rootkit.com, Blackstar, blackstar, LinkedIn, linkedin, Linkedin, Gamigo, gamigo, EHarmony, eHarmony, eharmony, Hellfire, hellfire, Whitefox, whitefox, Casio.cn, casio.cn, DamnSmallLinux, damnsmalllinux, Dhool, dhool, Gaming, gaming, FFGBeach, ffgbeach, Battlefield, battlefield, ABC
Adding dates	All possible dates having the form dd/mm/yy or mm/dd/yy are added (e.g. 20/05/74, 05/20/74).
Repeating special character groups	2-times and 3-times of "abc","abcd","qwer", "asdf","qaz","zxcv", "Abc", "ABC", "Abcd","AbCd" (e.g. ABCABCABC)
Repeating number groups	2-times and 3-times the numbers between 10-2100 are repeated (e.g. 959595)
Repeating birth years	The birth years between 1900 and 1999 are repeated 2-times and 3-times (e.g. 19101910, 195519551955).
Repeating symbols	The symbols (i.e. . ! , ; & | ( ) * + - < > = [ ] ^ ? $ % : # / @ _) are repeated 1-10 times (e.g. !!!,!!!!!, &&&&&&).
Repeating letters	The letters between a-z and the numbers between 0-9 are repeated between 1-15 times (e.g. aaaa, bbbbbbb, 33333333)

only unique pattern-based passwords. Finally, the output file of pbp-generator can be utilized for more efficient dictionary attacks.

B. The Cracking Tests

We used pbp-generator to generate a pattern-based dictionary file from the original rockyou password list which contains 14,344,399 unique passwords. pbp-generator generated a pattern-based dictionary file that contains 2,247,786,433 (circa 2.3 billion) unique passwords. The new dictionary file contains 156 times more passwords compared with the rockyou list.

Having two different password files (i.e. the original rockyou list and the generated pattern-based dictionary fi we performed dictionary attacks by using Hashcat tool [15]. In our analysis, we used real-life MD5 and SHA1 password hashes that were disclosed by different cyber-attacks and made publicly available [16] on the Internet.

We performed two parallel tests. In the fi t test, we checked how many password hashes can be cracked with the original rockyou password list. In the second test, we checked how many password hashes can be cracked by using our pattern-based password list generated by pbp-generator. As the success results and cracked password examples given in Table XVIII show, our patterns enabled many more additional hashes to be cracked. For example, ca. 577,000 Gamigo.com password hashes could be cracked with the help of the rockyou list. On the other hand, the pattern-based dictionary file could crack ca. 365,000 additional password hashes which could not be cracked with the rockyou list. Based on this result, 63% more passwords could be cracked with our patterns. Similarly, by eharmony.com analysis the pattern-based dictionary could crack ca. 28,000 additional passwords. This concludes that we could crack 150% more passwords compared with cracking with the rockyou list which could crack only ca. 18,500 password hashes.

C. Performance Analysis

Since the pattern-based dictionary contains many more passwords than the rockyou list, it takes longer to perform hash cracking with the pattern-based dictionary. The hash cracking tests were performed on a 64-bit machine with an Intel i5 dual core 3.2 GHz processor and 12 GB RAM. Hashcat was executed with 32 parallel-running threads. Testing with the rockyou dictionary took 37 seconds to complete the test for Gamigo.com having about 7 million password hashes. The same test took 8 minutes 59 seconds to complete when testing with the pattern-based dictionary.

V. RELATED WORK

Password security and cracking password hashes were extensively studied by many security researchers in the past. However, to the best of our knowledge, there is no other study which analyzes real-life patterns in detail, identifies several common password patterns and utilizes them to increase success rates of dictionary attacks as explained in this paper.

A. Pattern Analysis

Veras et al. [17] studied password patterns too, but they focus only on numbers and different date formats in passwords. They did not perform any password cracking benchmark test based on their identified patterns.

Wu [18] analyzed password security of a Kerberos realm containing slightly over 25 thousand users. They could crack a total of 2,045 passwords successfully by the end of the two-week experiment. The half of the guessed passwords

[1]All hashes of the resources were taken from http://www.adeptus-mechanicus.com/codex/hashpass/hashpass.php
[2]Diff column shows the number of password hashes that could be cracked only by using our pattern-based dictionary file.
[3]Success rate column shows how many percent of additional passwords could be cracked with the pattern-based dictionary file.

was from a dictionary. For the remaining half, they used the patterns prefix, suffix, capitalization and reversing. Comparing with our patterns, both their identified pattern set and benchmarking dataset are very limited.

B. Cracking Tests

Weir et al. [19] performed password cracking attacks against many real-life passwords including the rockyou database. They analyzed the passwords according to the NIST SP800-63 policy rules and showed that Shannon entropy as defined by NIST does not provide a good model to check security complexity of passwords. In their model, they compute the probability for a given password. Providing this, it is possible to blacklist passwords having the probability above a certain threshold since they are not secure against guessing attacks. We believe their model generates insecure passwords if we consider our pattern-based attacks. For example, their model suggests violin123 and !!password123 as strong secure passwords. But this is not correct. These passwords contain certain patterns. We show in this paper that such passwords can be easily cracked with pattern-based dictionaries. Stone-Gross et al. [20] took control of the Torping botnet which contained 297,962 unique username and password pairs. They did password cracking analysis by using john-the-ripper [21] in brute-force mode and could crack ca. 100 thousand passwords in 24 hours. Yan et al. [22] explain their empirical study which investigates the trade-off between security and memorability. They set up three different groups which chose their passwords freely or based on a mnemonic phrase. The last group was given a random password. In the end, they performed dictionary attacks to crack passwords of the study attendees. About 32% of the freely-chosen passwords could be successfully cracked. In their dictionary attack, they used replacement pattern as well, but in a very basic form. Weir et al. [23] created automatically a probabilistic context-free grammar based upon a training set of previously leaked passwords and used this grammar to generate word-mangling rules which were afterward used for password cracking tests. They were able to crack 28% to 129% more passwords than John the Ripper. Our approach achieves better results for certain datasets. For example, we could crack 151% more passwords in case of Eharmony.com dataset and similarly 239% more passwords in case of DamnSmallLinux dataset compared with their results. Moreover, their test dataset is very limited. One of their dataset contains 67,072 passwords and the other one contains 7,480 passwords. We used 15 different datasets and the Gamigo.com dataset contains alone more than 7 million passwords. Zhang et al. [24] presented a large-scale study of password expiration in practice. They provided an effi search algorithm framework for attacking future passwords from expired ones. They applied their search algorithm to a large, real-world data set for the analysis of password expiration and confirmed that password expiration is not an effective approach as expected.

C. Complexity Analysis

Imperva analyzed the complexity of the rockyou passwords and released a study [25]. According to their results, sixty percent of the passwords are quite insecure and contain only lowercase letters, uppercase letters or numeric values.

About thirty percent of the passwords have the length which is equal to or below six characters. They listed the most frequently used 20 passwords as well. "123456" is at the top in the list. This analysis shows only generic complexity results, but does not mention any patterns. Houshmand and Aggarwal [26] propose a new system which analyzes whether a user proposed password is weak or strong by estimating the probability of the password being cracked. They modify then the weak password to create a strengthened password as well. Some examples of weak and strengthened password are trans2 → %trans2, colton00 → 8colton00. This system is also insecure against pattern-based dictionary attacks. An attacker can delve into the details of this system, identify specific patterns used by this system and use these identified patterns to generate possible strengthened passwords. Stanekova and Stanek [27] evaluate several methods of choosing PIN against dictionary-based guessing attacks and discuss two methods for constructing easy to remember PIN words for randomly chosen PINs. Narayanan and Shmatikov [28] show how to reduce the size of password search space for dictionary attacks by using Markov modeling techniques. Mazurek et al. [29] performed an empirical study over the plaintext passwords of 25 thousand faculty, staff, and students at a research university. They found that some elements of the university population create more secure passwords than others. For example, computer science students make passwords more than 1.8 times as strong as the business school students. Comparing their contributions with ours, their focus is mainly the relation analysis of different categories like gender, college types, user types, etc. rather than password patterns. Jakobsson and Dhiman [30] built a model of passwords by using the Rockyou dataset. They parsed and scored passwords from five other datasets of disclosed passwords (i.e. Rootkit, Sony, Paypal, Justin Bieber fan web page and Porn web page datasets). They analyzed then the usage of various rules in the datasets. Their analysis showed the average number of components per password in the different datasets. As a result, they found out that Justin Bieber dataset has the highest average number of word components compared with the other datasets. Kelley et al. [31] studied the impact of different password policies on password strength. They investigated mainly the resistance of passwords created under different policies and the performance of guessing algorithms under different training sets.

D. New Password Schemes

Forget et al. [32] proposes a password creation scheme based on Persuasive Technology [33]. This scheme inserts or replaces randomly fixed number of characters in a user chosen password. As explained in this paper, inserting or replacing characters are typical patterns which can be misused to guess passwords successfully. Xiao et al. [34] propose some password mechanisms in which a user can choose a virtual password scheme ranging from weak security to strong security. The proposed schemes provide several system recommended functions like flipping one digit in the password, reversing bits of the password, adding an additional digit/character at a fixed place, etc. We showed that such functions can be attacked since their results contain certain patterns.

VI. Mitigation Methods

The following mitigation methods can be suggested in order to minimize the risks from patterns and protect users against unauthorized access to their accounts.

One possible solution can be that users exploit secure password managers (SPM) to store their passwords. SPMs generate unique, random and complex passwords without any pattern, store them within a database and store the database in an encrypted form (e.g. AES-256) on file systems. In order to decrypt the database and retrieve the passwords, a master secure password must be provided by users. In addition, some SPMs ask users to provide a physical file which is generated randomly during the setup phase of the password database creation. Providing this, users generate secure passwords for each service they use with the help of their SPM and do not need to memorize them. They just need to memorize the master password and protect the physical file against unauthorized access. It is in this case important that the master password is complex, randomly generated and contains no pattern. But it is not a problem for users to memorize a single complex password and remember it later. Furthermore, some SPMs offer smart-card authentication.

Another solution can be two-factor authentication. Today authentication systems should not depend only on knowledge of username-password pairs, especially for critical applications like email, online banking or e-commerce. A new authentication factor based on what we possess (e.g. hardware token, smart-card) or who we are (e.g. fingerprint) should be additionally checked during authentication process. As examples, online banking applications benefit today tamper-resistant hardware tokens and similarly some online services like Google Mail, Twitter, Wordpress etc. support software tokens that are sent over SMS or generated by a native mobile app (e.g. Google Authenticator).

Considering the pattern risks, it is vital to revise current password authentication systems as well. They normally check if a user-given password is a dictionary word or not. If it is a dictionary word, it is black-listed and rejected. The user is asked to choose a non-dictionary password. This existing feature should be extended to cover passwords with patterns. They can let pbp-generator create a pattern-based dictionary file from their current dictionary file and afterward check if users enter passwords which exist within the pattern-based dictionary file.

Academic researchers focusing on password security and authentication systems should take patterns into consideration and propose solutions accordingly. The related academic works from the past should be re-evaluated by considering the risks caused by patterns. Security awareness trainings held especially for non-security experts should take patterns into consideration as well. Attendees should be informed about the patterns and warned not to use pattern-based passwords.

VII. Conclusion

Weak passwords are critical threats for authentication systems. Seizing password hashes, especially unsalted hashes,

attackers can use different attack techniques (i.e. brute-force, dictionary, rainbow-tables) to crack hashes and reveal plaintext passwords. Security experts try to establish security awareness for strong passwords. In addition, authentication systems enforce password policies to fulfill complexity rules. Being forced to use strong passwords, people tend to use similar patterns when choosing their "strong" passwords. But such patterns endanger security of passwords.

In this paper we explain how frequently used patterns can be identified and misused to generate pattern-based password dictionaries. These common patterns can be afterward exploited to crack more password hashes compared with traditional dictionary attacks. In order to identify common password patterns, we performed both manual and automated analysis on a large set of leaked real-life passwords of rockyou.com gaming portal. After identifying the patterns, we developed a software tool, namely the pbp-generator, which creates many pattern-based passwords from a given traditional dictionary. We utilized the generated pattern-based dictionary to perform cracking tests against real-life leaked password hashes from 15 different datasets. According to the test results, we could crack with pattern-dictionaries many more password hashes, which cannot be cracked by using the rockyou password list. From this perspective, our proposed pattern-based attacks enhance dictionary attacks and can be considered as the new generation of dictionary attacks. It can especially help forensic investigators for more efficient password cracking compared with the existing techniques.

ACKNOWLEDGMENT

The author would like to thank Necati Erşen Şişeci, M. Oğuzhan Topgül, M. Oğuzhan Külekçi and Yalçın Çakmak who provided valuable comments on drafts of this article.

REFERENCES

[1] L. O'Gorman, "Comparing passwords, tokens, and biometrics for user authentication," *Proc. IEEE*, vol. 91, no. 12, pp. 2021–2040, Dec. 2003.

[2] (2011). *PlayStation Network Hack: Why it Took Sony Seven Days to Tell the World.* [Online]. Available: http://www.theguardian.com/technology/gamesblog/2011/apr/27/playstation-network-hack-sony

[3] (2009). *RockYou Hack Compromises 32 Million Passwords.* [Online]. Available: http://www.scmagazine.com/rockyou-hack-compromises-32-million-passwords/article/159676/

[4] (2013). *Software Company Tom Sawyer Hacked, 61,000 Vendors Accounts Leaked*. [Online]. Available: http://www.databreaches.net/software-company-tom-sawyer-hacked-61000-vendors-accounts-leaked/

[5] (2013). *Hackers Leak Data Allegedly Stolen from Chinese Chamber of Commerce Website.* [Online]. Available: http://news.softpedia.com/news/Hackers-Leak-Data-Allegedly-Stolen-from-Chinese-Chamber-of-Commerce-Website-396936.shtml

[6] *LinkedIn Hack*. [Online]. Available: http://en.wikipedia.org/wiki/2012_LinkedIn_hack, accessed Apr. 22, 2015.

[7] *SQL Injection*. [Online]. Available: https://www.owasp.org/index.php/SQL_Injection, accessed Apr. 22, 2015.

[8] *Password Storage Cheat Sheet*. [Online]. Available: https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet, accessed Apr. 22, 2015.

[9] *Brute-Force Attacks*. [Online]. Available: https://www.owasp.org/index.php/Brute_force_attack, accessed Apr. 22, 2015.

[10] V. Goyal, V. Kumar, M. Singh, A. Abraham, and S. Sanyal, "CompChall: Addressing password guessing attacks," in *Proc. Int. Conf. Inf. Technol., Coding Comput. (ITCC)*, Apr. 2005, pp. 739–744.

[11] P. Oechslin, "Making a faster cryptanalytic time-memory trade-off," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 2729, D. Boneh, Ed. Berlin, Germany: Springer-Verlag, 2003, pp. 617–630.

[12] G. Fragkos and T. Tryfonas, "A cognitive model for the forensic recovery of end-user passwords," in *Proc. 2nd Int. Workshop Digit. Forensics Incident Anal. (WDFIA)*, Aug. 2007, pp. 48–54.

[13] C. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 1948.

[14] *SkullSecurity Passwords*. [Online]. Available: https://wiki.skullsecurity.org/Passwords, accessed Apr. 22, 2015.

[15] *Hashcat—Advanced Password Recovery Practices*. [Online]. Available: http://hashcat.net, accessed Apr. 22, 2015.

[16] *Hashdumps and Passwords*. [Online]. Available: http://www.adeptus-mechanicus.com/codex/hashpass/hashpass.php, accessed Apr. 22, 2015.

[17] R. Veras, J. Thorpe, and C. Collins, "Visualizing semantics in passwords: The role of dates," in *Proc. 9th Int. Symp. Vis. Cyber Secur. (VizSec)*, 2012, pp. 88–95.

[18] T. Wu, "A real-world analysis of Kerberos password security," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 1999. [Online]. Available: https://www.gnu.org/software/shishi/wu99realworld.pdf

[19] M. Weir, S. Aggarwal, M. Collins, and H. Stern, "Testing metrics for password creation policies by attacking large sets of revealed passwords," in *Proc. 17th ACM Conf. Comput. Commun. Secur.*, New York, NY, USA, 2010, pp. 162–175.

[20] B. Stone-Gross *et al.*, "Your botnet is my botnet: Analysis of a botnet takeover," in *Proc. 16th ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2009, pp. 635–647.

[21] *John the Ripper Password Cracker*. [Online]. Available: http://www.openwall.com/john/, accessed Apr. 22, 2015.

[22] J. Yan, A. Blackwell, R. Anderson, and A. Grant, "Password memorability and security: Empirical results," *IEEE Security Privacy*, vol. 2, no. 5, pp. 25–31, Sep./Oct. 2004.

[23] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in *Proc. 30th IEEE Symp. Secur. Privacy (SP)*, May 2009, pp. 391–405.

[24] Y. Zhang, F. Monrose, and M. K. Reiter, "The security of modern password expiration: An algorithmic framework and empirical analysis," in *Proc. 17th ACM Conf. Comput. Commun. Secur. (CCS)*, 2010, pp. 176–186.

[25] M.Muthamil Jothi, A.Nancy, V.Manjula, R.Muthu Veni, S.Kavya, Christo Ananth, "Efficient message forwarding in MANETs", International Journal of Advanced Research in Management, Architecture, Technology And Engineering (IJARMATE), Vol. 1, Issue 1, August 2015, pp:6-9

[26] S. Houshmand and S. Aggarwal, "Building better passwords using probabilistic techniques," in *Proc. 28th Annu. Comput. Secur. Appl. Conf. (ACSAC)*, 2012, pp. 109–118.

[27] L. Staneková and M. Stanek, "Analysis of dictionary methods for PIN selection," *Comput. Secur.*, vol. 39, pp. 289–298, Nov. 2013.

[28] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *Proc. 12th ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2005, pp. 364–372.

[29] M. L. Mazurek *et al.*, "Measuring password guessability for an entire university," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 173–186.

[30] A.Nasrin Banu, M.Manju, S.Nilofer, S.Mageshwari, A.Peratchi Selvi, Christo Ananth, "Efficient Energy Management Routing in WSN", International Journal of Advanced Research in Management, Architecture, Technology And Engineering (IJARMATE), Vol. 1, Issue 1, August 2015, pp:16-19

[31] P. G. Kelley *et al.*, "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2012, pp. 523–537.

[32] A. Forget, S. Chiasson, P. C. van Oorschot, and R. Biddle, "Improving text passwords through persuasion," in *Proc. 4th Symp. Usable Privacy Secur. (SOUPS)*, 2008, pp. 1–12.

[33] B. J. Fogg, "Persuasive technology: Using computers to change what we think and do," *Ubiquity*, Dec. 2002

[34] Y. Xiao, C.-C. Li, M. Lei, and S. V. Vrbsky, "Differentiated virtual passwords, secret little functions, and codebooks for protecting users from password theft," *IEEE Syst. J.*, vol. 8, no. 2, pp. 406–416, Jun. 2014.