

# Visualizing Keyboard Pattern Passwords

Dino Schweitzer, Jeff Boleng, Colin Hughes, Louis Murphy

United States Air Force Academy

## ABSTRACT

Passwords are a fundamental security vulnerability in many systems. Several researchers have investigated the tradeoff between password memorability versus resiliency to cracking and have looked at alternative systems such as graphical passwords and biometrics. To create stronger passwords, many systems enforce rules regarding the required length and types of characters passwords must contain. Another suggested approach is to use passphrases to combat dictionary attacks. One common “trick” used to remember passwords that conform to complex rules is to select a pattern of keys on the keyboard. While appearing random, the pattern is easy to remember. The purpose of this research was to investigate how often patterns are used, whether patterns could be classified into common categories, and whether those categories could be used to attack and defeat pattern-based passwords. Visualization techniques were used to collect data and assist in pattern categorization. The approach successfully identified two out of eleven passwords in a real-world password file that were not discovered with a traditional dictionary attack. This paper will present the approach used to collect and categorize patterns, and describe the resulting attack method that successfully identified passwords in a live system.

**KEYWORDS:** Visualization, security, passwords.

**INDEX TERMS:** K.6.5 [Management of Computing and Information Systems]: Security and Protection—Authentication, Unauthorized access

## 1 INTRODUCTION

Passwords are still the most widely used technique for system user authentication. The need for strong passwords in today’s age of ever faster computers with an explosion of parallel processing capability is more important than ever [1]. In a 2004 study, Yan, et. al. found that about one third of computer users chose weak or crackable passwords [2]. Moreover, one vulnerable password could create vulnerabilities across many systems, since people use the same password for, on average, six accounts [3]. Typically these passwords are found using a dictionary attack—that is, a program that hashes a list of common words, phrases, and other likely passwords until it finds a match. Modern day computers can attempt millions of possible passwords per minute. In an effort to “force” users into selecting strong passwords, system administrative policies are often implemented with complex rules. These rules require special characters, minimum password lengths, and even forbid dictionary words.

HQ USAFA/DFCS, 6G-101, USAF Academy, CO 80840  
dino.schweitzer@usafa.edu

An example of such password rules would be:

1. Minimum password length of 12 characters
2. Must contain 2 special characters
3. Must contain 2 numbers
4. Must contain 2 upper case letters
5. Must NOT contain any dictionary words

It is common for users to struggle to create passwords that meet these rules. Furthermore, it is even more difficult to create an easily remembered password that meets these rules.

Whether as a result of stringent password composition rules, or just the pursuit of a potentially strong password, many users have resorted to alternative passwords such as passphrases and keyboard patterns for their password [4][5]. An example of such a pattern-based password is 1qaz!QAZ2wsx@WSX. While this seemingly random, 16-character password meets the above rules and appears quite strong, when the sequence of keystrokes is analyzed, its pattern becomes immediately apparent. Figure 1 is a visualization of this password.

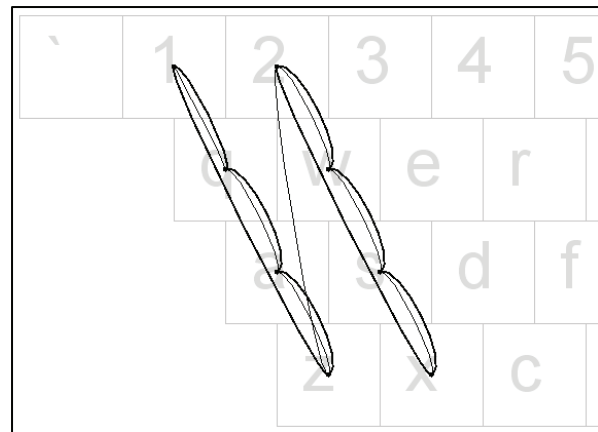


Figure 1. Keyboard visualization of 1qaz!QAZ2wsx@WSX.

It is clear the user chose an easy-to-remember, but obvious keyboard pattern starting with the “1” key down to the “z” key, the same sequence with shift pressed, and then the repeated pattern on the diagonal column from “2” to “x”. This paper outlines a visualization technique and accompanying development of heuristics that allow seemingly strong pattern-based passwords to be easily recognized and efficiently attacked.

## 2 COMMON PASSWORD ATTACKS

There are a number of common ways to break passwords. These techniques range from social engineering, phishing, shoulder surfing, keyboard logging, and dictionary based attacks, to brute force attacks. The goal of strong passwords is to defeat dictionary based attacks and require brute force attacks that are

computationally infeasible. Password cracking systems have become increasingly sophisticated and complex. “John the Ripper” is one well-known password cracking tool [6]. Not only does “John” allow dictionary and brute force attacks, it also has an advanced rules engine that can automatically create password variations of common dictionary words. It is common for users to alter their simple dictionary passwords by substituting special characters and numbers in order to transform them into stronger passwords. An example would be instead of using the name of your favorite dog, fido, as an easily broken password, simply alter the characters to something like f!d0. This greatly increases the search space required and seems to make this a non-dictionary word. However, tools like “John” can easily create dictionaries with such simple variations that render such password composition tricks nearly useless. As a consequence of these advances in password cracking, security-minded users are motivated to create more complex passwords. In an effort to easily remember their password, while also avoiding such dictionary based variation attacks, many users have resorted to passwords based on keyboard patterns.

### 3 OUR APPROACH

Our research focused on the ability to recognize keyboard patterns. We used the following approach:

- Collect a dataset of passwords, including some that are known to be pattern-based
- Develop a visualization technique to analyze passwords for common pattern elements
- Develop heuristics based on recognized patterns
- Create a password file by generating a dictionary from the pattern heuristics
- Apply a password cracking tool and the pattern dictionary against a real password file

#### 3.1 Collecting Passwords

Our goal was to collect a large, unbiased sample of possible passwords. We could not do this simply by analyzing an existing password file since we would not know which passwords were intended to be patterns. To address this challenge we created a web-based password tutorial and asked our freshman computer science class to complete it. The tutorial taught concepts such as password strength, cracking techniques, etc. It was highly interactive and encouraged active participation. We asked the students to enter candidate passwords and gave them feedback on whether it met the required composition rules as well as indicating the relative password strength. All passwords the students experimented with were captured and logged into a password database which we later used for analysis.

One page of the tutorial explained the concept of using keyboard patterns to generate passwords and asked them to imagine and enter a password based on a keyboard pattern they created. We used visualization to illustrate the pattern to the students in real time as they entered them and then allowed them to re-animate it. This visual feedback motivated them to experiment and provided multiple candidate pattern-based passwords. An image of the tool used is shown as Figure 2. Students entered patterns twice, once as a random pattern as shown in Figure 2, and once with a pattern that met the specified length and character constraints.

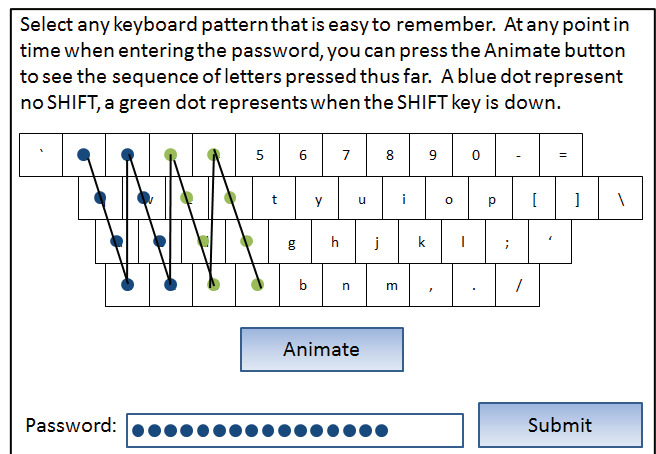


Figure 2. Passwords Web Lab for keyboard patterns

Using the tool we collected over 250 unique pattern-based passwords from 161 distinct users for analysis. Additionally, we logged over 500 “random” passwords that could be evaluated for patterns even though patterns were not explicitly requested when these passwords were entered.

#### 3.2 Visualizing Password Patterns

Our next step was to examine the large number of pattern-based passwords collected for possible common features. Simply drawing lines connecting the sequence of keys on a graphical keyboard was insufficient. Lines overlapped when keys were in the same row, there was no easy way to distinguish multiple presses of the same key, and the sequence of key presses was confusing. We needed to develop ways to visualize when the shift key was used, when a single key was pressed multiple times in sequence, and when a string of keys were pressed multiple times in sequence. Our first approach, similar to the Web Lab that was used to collect passwords, was to animate and “playback” the password sequence. This allowed the patterns to be recognized but was not easy to visualize and scan large numbers of passwords quickly.

To improve the recognition of patterns when comparing multiple passwords, the following drawing rules were used:

1. Connect sequentially pressed keys with an arc.
2. If the shift key is pressed, increase the weight of the connecting arc.
3. Multiple presses of the same key are shown as tight loops which create a flower-like pattern.
4. Repetition of a key sequence is shown using offset arcs.
5. Repetition of same-key presses are shown with offset (or concentric) petals.
6. Arcs are always drawn in clockwise order to indicate the order in which the keys were pressed.

Figure 3 demonstrates each of these drawing rules. A few things to notice are the bold lines on the right of the figure showing the use of the shift key. Also, it is apparent the “d” and “t” keys were pressed multiple times. In this case the four “d” key presses are shown by the incoming arc and the “flower” with three petals. Similarly, the “t” key has offset petals showing it was pressed more than four times. The repeated key sequence “qwqwqw” is visually obvious with the offset arcs. Finally, because arcs are always drawn clockwise you can determine the sequence of key presses without the need for arrowheads.

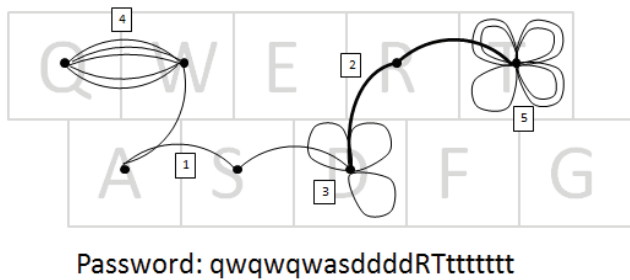


Figure 3. Password pattern visualization and drawing rules

Figure 4 shows examples of individual password patterns collected in the dataset. When a password was not created as a pattern, as shown in the fourth example, it is visually obvious. While interesting and useful for looking at single patterns, we wanted to identify elements of patterns that were common across several passwords. In order to quickly identify common elements, the tool can display a table of multiple password patterns (4, 9, or 16 passwords at a time). The user can page back and forth through the database looking at several passwords simultaneously. This is illustrated in Figure 5. As a result of this visualization, it was apparent that some type of pattern appeared in almost 20% of the random passwords, even without prompting the user to enter a keyboard pattern.

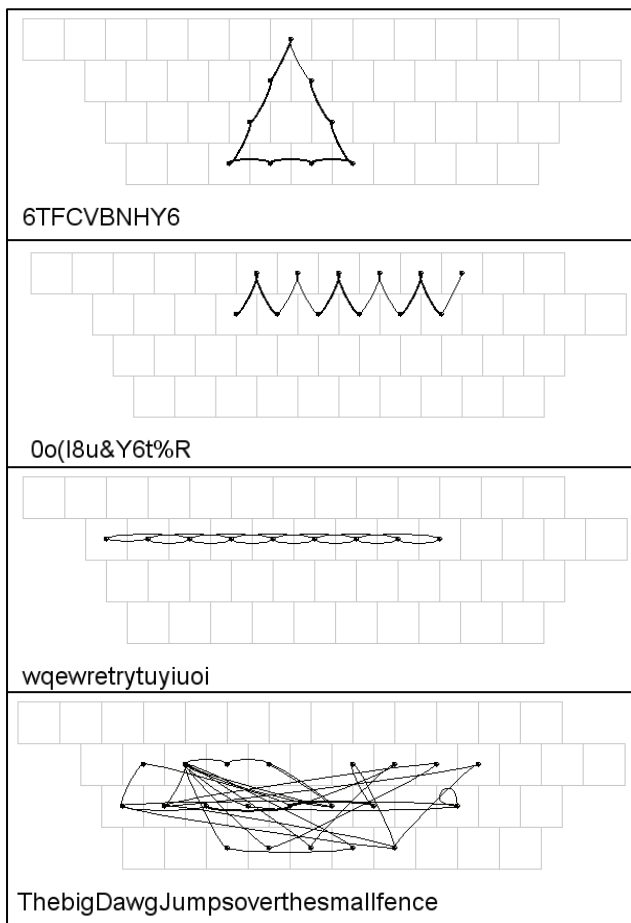


Figure 4. Individual passwords patterns

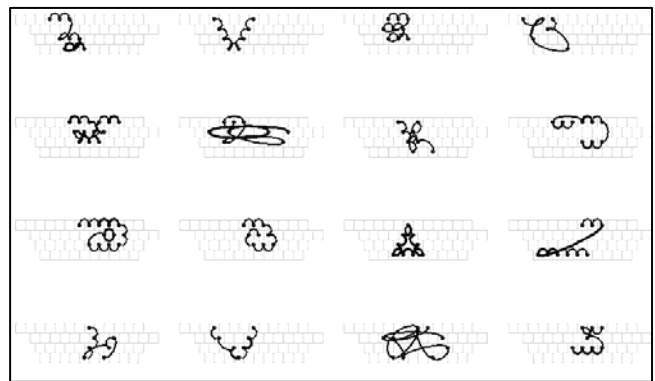


Figure 5. Visualizing multiple passwords for patterns

### 3.3 Finding Common Elements

Using the tool shown in Figure 5 we visually studied our database of passwords. The goal was to identify common elements that appeared in several patterns. We found several frequently used types of patterns in our dataset that are summarized with examples in Table 1. Based on these common elements identified visually, we wrote code to perform a more exact count of how many passwords in our dataset contained each element type. Figure 6 shows the results. The three datasets compared are 1) passwords that were entered with no prompting for a pattern, 2) passwords where students were prompted to enter any pattern, and 3) passwords where the pattern needed to meet the specified constraints on length and character sets. Not surprisingly, two contiguous keys appear most often, as they naturally occur in normal English words. Three consecutive keys in a row appeared in over half of the “any pattern” dataset, and over a quarter of patterns that were constrained, but not very often in random passwords. Because “threes” is a subset of “snakes” the “snakes” category is also high. If you remove the “threes” from the “snakes” they appear in about 10% of all password types. Surprisingly (at least to us), the flower petal pattern of doubles and triples did not appear very often. In fact, a “single petal” appeared in only 10% of the “no pattern” passwords and zero times when students were prompted to enter patterns. A pattern of particular interest is the “grouped 2/3/4’s”. These patterns are technically not “snakes” because the start of a new set is not necessarily contiguous to the end of the previous set. However, the pattern appeared in over 15% of the random pattern dataset. This particular pattern was a promising pattern to attempt to crack.

### 3.4 A Prototype Pattern-Based Password Crack

The next step of the project was to determine ways of taking advantage of the information found in pattern examination to try to crack pattern passwords. One can imagine a tool specifically designed to generate random patterns based on the elements to generate and check passwords against these patterns. We chose a simpler approach to demonstrate one of the patterns we identified. We algorithmically generated pattern passwords and captured the output in a dictionary for use in the standard attack program.

“John” has several options for generating password variations. One option is to look at contiguous keys, or “snakes” in our pattern categorization venacular. This option will recognize the categories “twos”, “threes”, “fours”, and “down the row” as types of snakes. A common pattern that it will not discover, however, is the “grouped 2/3/4’s”. This is the pattern we decided to try and attack.


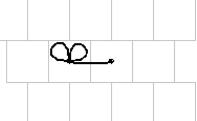
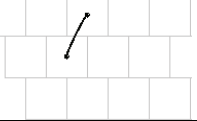
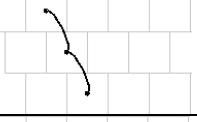

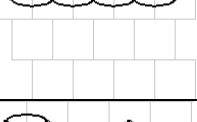

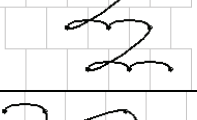

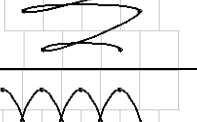
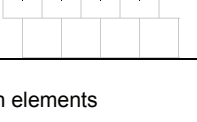
Pattern Name	Description	Example
Doubles	Same key pressed twice in succession	
Triples	Same key pressed three times in succession	
Twos	Two keys in a continuous line	
Threes	Sets of three keys in a continuous line	
Fours	Sets of four keys in a continuous line	
Down-the-row	Five or more keys in one row	
Snake	Sequence of contiguous keys	
Grouped 2/3/4's	Sets of 2's, 3's, 4's offset by row or diagonal	
Split Snake	Two discontinuous snake parts	
Reflected	Sequence of mirrored keystrokes	
Zig-zag	Alternating contiguous keys from two rows	

Table 1. Common keyboard pattern elements

Using the pattern heuristics described above for the “grouped 2/3/4’s” pattern, a set of rules was used that generated all possible passwords that met this pattern. We used these rules to exhaustively generate a dictionary consisting of 8-12 character passwords. The minimum and maximum lengths were arbitrarily chosen based on common password lengths observed. Longer passwords can be easily generated with a resulting increase in dictionary size.

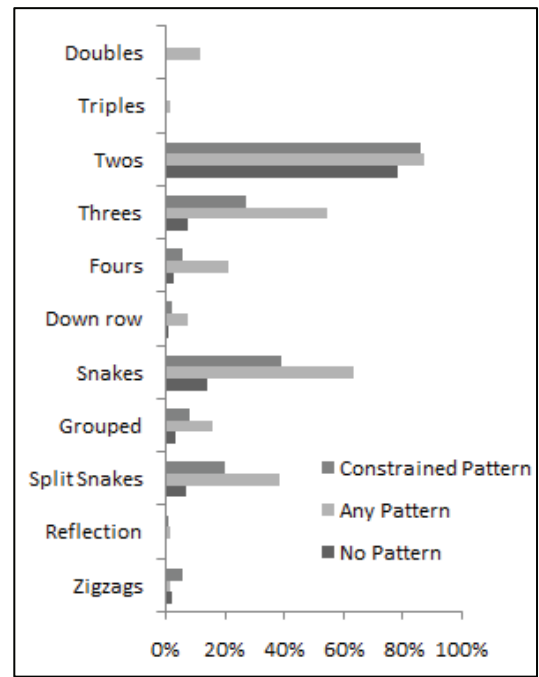


Figure 6. Frequency of elements in password datasets

The password generation tool created passwords with the following structure (listed in priority order):

1. Meets locally enforced complexity rules
2. Password parts go in the same direction (individual sets of 2, 3, or 4 keys in a row)
3. Password parts are the same length
4. Password parts are either all shifted up, or all not
5. Password parts go left to right, or top to bottom
6. Password parts are only 1 space off on the keyboard

Table 2 shows visual examples of the types of password patterns generated using these rules.

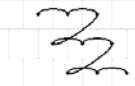

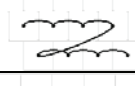
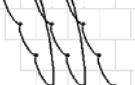
Number of 1-key shifted sub-groups	Number of adjacent keys in each sub-group	Example
3	3	
4	3	
2	4	
3	4	

Table 2. Sample 8-12 character “grouped” patterns

The resultant dictionary had almost half a million entries, but this is a relatively small number compared to the full search space. Assuming a QWERTY keyboard has 94 characters, the total

search space is on the order of  $4.8 \times 10^{23}$  ( $94^8+94^9+94^{10}+94^{11}+94^{12}$ ) possible password combinations. This is far too large for any brute force attack to be computationally feasible. “John,” when using the pattern generated password file of nearly 500,000 entries, was able to check each password against the entire dictionary in just a few seconds. This speed was enhanced by turning off the default rules which try variations on each dictionary password (e.g. fido, F1d0, f!Do, etc.).

### 3.5 Results

To test the results of our approach, we initially attempted to crack the passwords in the entire dataset collected for this project (both patterns and random) using “John” and the dictionary generated above based on the “grouped” pattern. This resulted in the discovery of over 20% of the passwords. A more meaningful test is to run it against a “real” password file. After a great deal of negotiation and promises of anonymity, we obtained a small password file from a live production system at our institution that contained 11 “strong” password hashes. Although the file was small, with the default dictionary, “John” quickly defaulted to brute force mode and did not discover any passwords in over 18 hours of run time. Using the dictionary generated in accordance with the pattern heuristics we developed, “John” was able to discover two of the 11 passwords (18%) in less than a second. Since these passwords were obtained from a production system, this points to a serious weakness in the use of pattern-based passwords.

### 4 FUTURE WORK

Our approach was limited to a subset of pattern-based passwords. It would be possible to include a greater portion of patterns by relaxing one or more of the rules we used to generate the dictionary outlined above. Alternately, a dictionary of 2, 3, and 4 character fragments could be used with regular expressions to combine them and check for complexity. This approach would be somewhat slower than ours, but still much more effective than a brute force attack.

### 5 CONCLUSION

Given the high reliance on passwords as a primary means for authentication and the commonly used administrative constraints placed on users to attempt to enforce “strong” passwords, pattern-based passwords are increasingly common. Our large dataset and our small production password file both indicate pattern-based passwords are common and they can be efficiently cracked. Through visualization we recognized common pattern characteristics. Using these characteristics we developed a set of heuristics to generate a pattern-based password dictionary that dramatically reduced the time taken by a common password cracking tool to discover pattern-based passwords. The fact that this approach was able to identify two out of eleven passwords in a live password file that were not discovered with a traditional dictionary attack indicates it is a real problem that needs to be addressed for greater password security.

### REFERENCES

- [1] J. Bengtsson. Parallel Password Cracker: A Feasibility Study of Using Linux Clustering Technique in Computer Forensics. *Proceedings of the Second international Workshop on Digital Forensics and Incident Analysis WDFIA* (2007).
- [2] J. Yan, A. Blackwell, R. Anderson, and A. Grant. Password memorability and security: empirical results. *IEEE Security & Privacy* 2, 25-31 (2004).

- [3] D. Florencio and H. Cormac. A Large-Scale Study of Web Password Habits. *Proceeding. of 16th international world wide web conference*. Banff, Canada, (2007).
- [4] A. De Luca, R. Weiss, and H. Hussmann. passShape – Stroke based Shape Passwords. *Proceedings of the 19th Australasian Conference on Computer-Human interaction: Entertaining User interfaces, OZCHI '07*, vol. 251, (2007).
- [5] C. Kuo, S. Romanosky, and L. F. Cranor. Human selection of mnemonic phrase-based passwords. *Proceedings of the Second Symposium on Usable Privacy and Security SOUPS '06*, vol. 149, (2006).
- [6] <http://www.openwall.com/john/>