IMPLEMENTING RISK SCORE TO PROTECT FROM ANDROID PATTERN LOCK ATTACKS

Yasir Al-Qaraghuli and Caroline Hillier

Department of Computer Science, University of Guelph, Ontario, Canada

ABSTRACT

Cyberattacks on Android devices have increased in frequency and commonly occur in physical settings with shoulder surfing and brute-force attacks. These attacks are most common with devices secured by the pattern lock mechanism. This work aims to investigate the various methods that increase the security of Android lock patterns. Research showed that these pattern lock screens are especially vulnerable due to users employing a set of common lock patterns. We propose a pattern-matching algorithm that recognizes these common lock patterns and increases the Risk Score if these passcodes are attempted. The blocking of common passcodes, and identification during the unlocking, reduces the risk of the aforementioned threats to device security. The algorithm we implemented succeeds in deterring users from configuring their devices with commonly used patterns. Overall, our algorithm achieves advanced security compared to current systems by detecting unusual inputs and locking the device when suspicious activity is detected. Our test results show 80% satisfaction from human test subjects when settings the passcode. The algorithm eliminates the use of commonly used patterns and 79% acceptance using our proposed algorithm and blocks access to the device depending on the accuracy score. The proposed algorithm shows remarkable success with limiting brute-force attacks as it proves effective in denying common passcodes.

KEYWORDS

Android device, Lock pattern, Brute-force, Shoulder-surfing, Pattern Recognition.

1. INTRODUCTION

Over 2.5 billion users worldwide use Android mobile devices [1]. These devices hold a great deal of valuable personal information such as addresses, passwords, and personal documents [2]. Because of the value of this data, there can be severe repercussions if a device's security is compromised.

Research has shown that users created passcodes to fall into a small set of reoccurring patterns [2]. Complex lock patterns can be hard to memorize, and certain nodes are easier to reach while holding a device [3][4]. There is a necessary balance for users to create a unique pattern that they can remember and repeat frequently, but there will always be some human error.

Many ways that forgetful users or attackers can bypass the lock screen, though these processes are time consuming or result in data loss [5]. Because of these long processes, it is ideal for attackers to directly attack the lock screen.

Brute-force attacks occur when an attacker attempts passwords on a device with the intent to gain unauthorized access [6]. Shoulder surfing attacks occur when a criminal physically positions

David C. Wyld et al. (Eds): ICAIT, CBIoT, WiMo, CRYPIS, ICDIPV, CAIML, NLCA - 2022
pp. 103-112, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.121208

them self in a way where they can observe device passcode entry of their target [7], as shown in (Figure 1). We found that these threats are an exceptionally severe problem with Android devices as pattern locks are amazingly easy to a brute-force attack (Figure 2). Prior research showed that six dot-length Android pattern attacks with a single shoulder surfing observation had a 65% success attack rate, which increased to 79.9% with multiple observations by the attacker [8]. Our work will investigate several ways of identifying attacks and blocking them. We propose a *Risk Score* factor that evaluates the user based on their input and determines if the entry is a brute force attack or if the user made an input error. We have implemented this risk percentage system to allow the user to choose a risk allowance on their device. Once the user reaches the predetermined risk allowance with unsuccessful pattern attempts, they would be locked out and require additional verification.

Figure 1. Diagram of shoulder surfing

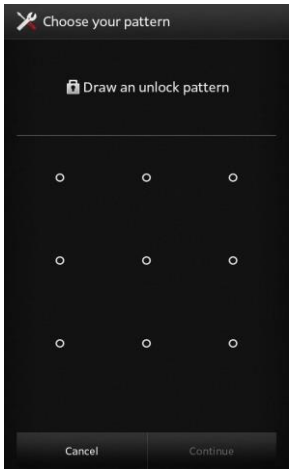

Figure 2. The famous android lock pattern

2. RELATED WORK

There has been some research regarding shoulder surfing, but the work typically focuses on the prevention, not the actual attack performance [8].

Previous studies have shown that having a six-digit pin over a four-digit (similar complexity) decreases shoulder surfing attack success from 64% to 11% [8]. The way that the user holds their phone and angle of observation also alter the attacker's success rate [8].

Other methods have utilized biometric trends in tools for the security of mobile devices. Instead of a passcode, researchers recorded users' physical traits to secure the device [3]. The biometric characteristics included keystrokes and touch dynamics [3].

Utilizing the sensors that track these characteristics can also assist in preventing attacker success, though it does lead to limitations such as having a friend use the device.

Many security professionals have aimed to investigate the vulnerabilities surrounding lock pattern systems. A computer vision algorithm showed that the analysis of fingertip motions often leads to successful shoulder surfing to brute force attacks [9]. Research showed that an attacker using fingertip analysis successfully guessed passwords in five attempts, with a 95% success rate [9].

The current methods for device locking are not sufficient and remain vulnerable [10].

Researchers have tested using an efficacy meter to enforce stronger user-created passwords [9]. The study compared the security of pattern selections of users seeing the meter and those that did not. The passwords of the participants that did not see the meter could be guessed in 16 guesses, while those that did see the meter needed 48 attempts to guess correctly [11].

Researchers have studied the implementation of blocklists to deter users from choosing the most common passcodes [1]. They found that the block list caused frustration and additional time from the user but overall reduced attack success rate [1]. 28 Android unlock patterns were identified as the most frequently used [1]. The authors describe the frequently used patterns as unsafe because they will likely be used in a brute force attack [1].

To better Android device security, we propose a system that identifies common patterns and lists them as a security risk. Our system both denies the user from using these common patterns and monitors for an attacker trying to access the system by entering these patterns. Our work aims to provide further protection for Android devices that implement a lock pattern system. Because we are also developing solutions for the security of Android lock patterns, we will use the database established in this work [1].

3. PROPOSED SOLUTION

Our system utilizes a risk scoring system to determine if the device should lock after each pattern lock attempt. The owner will set a lock pattern during setup, which is acceptable if it is not a common pattern [1]. The user will also establish their preferred risk acceptance level after an explanation of what the acceptance level means.

Our system uses a pattern matching algorithm to identify whether an inputted pattern matches a common pattern.

Since the user cannot set a common pattern as their passcode, the attempt of these patterns is a high-risk action.

Pattern matching also compares the current unlock attempt to the previous entry. Sequential unlock attempts with high variation signify that the user attempting to unlock the device does not know the passcode and is trying various patterns to unlock the phone. This behaviour indicates that the user unlocking the device is likely an attacker, thus increasing the *Risk Score*.

Actions deemed high risk will significantly raise the *Risk Score*, while actions that are low risk will marginally raise the *Risk Score*.

This ranking means that a series of minimal risk unlock attempts will gradually lock the user out, but a series of high-risk unlock attempts will lock the device more quickly.

When the device becomes locked, it may be subject to a time penalty or connection to a home device to unlock. The time penalty security measure has low efficacy if the attacker has the device and can resume their brute force after waiting a small amount of time. Increasing the lockout time may be effective against attackers but inconvenient for users. Similarly, requiring connection to a home device would be most effective as the attacker would not have access to the device but would be inconvenient for a user not near their home.

A solution for the locked-out device (Figure 3) is to have a recovery email tied to the account, shown in Figure 4. Device owners would receive the recovery email immediately after device lockout. This email system gives legitimate users an unlock link that lets them continue to unlock their devices. Attackers would not have access to the email link, meaning the device would remain locked. If a user is not currently using the device, this email could indicate that their device is being attacked. The recovery feature was implemented following user testing. In testing, we received feedback regarding "How long would [a user] have to wait until [they] can try again?" This email system complements our pattern detection system as an extra layer of security to keep brute-force attackers out.

Our algorithm provides improved security compared to previous research done on Android lock patterns. Creating a cohesive interface that blocks the use of commonly used patterns [1] and uses a *Risk Score* where users can enforce their security preference, provides strengthen security for the device locking system. Uniting the proposed security concepts allows for insight to real world integration.

Figure 3. Locked out user after multiple attempts

Figure 4. Email authentication to gain access back to device

4. METHODOLOGIES

4.1. Development

We chose to use Python to create both our front-end interface and our back-end code library. To create the Graphical User Interface, the *tkinter* interface package was used. The *smtplib* library was used to send out verification codes to user recovery email addresses.

4.1.1. Back-End Development

The 28 common patterns defined in [1] serve as the database of common passwords for this project. To allow for machine identification of the android patterns, we converted them into arrays by assigning each node with a number and then representing patterns as arrays. Pattern nodes are represented with the digits one through nine, with one being the node in the top left and nine in the bottom right. Zeroes padded the length of the array. Using these arrays, we could then perform pattern-matching by analysing the positions of node values and calculating how similar the arrangements were. We then created a library of back-end functions that could be called from our front-end code.

4.2. Pattern-Matching Algorithm

To perform the pattern-matching, we compared patterns based on their similarity to the identified common passcodes identified in [1]. Pattern location will be represented as numbers in a three-by-three grid, which can be seen in figure 6 [12]. Each pattern in the input would contribute to the *Risk Score*. If the node value appears in the same position as the correct pattern, one point is added to the *Risk Score*. If the node value is in the correct pattern but in different locations, half a point is added to the *Risk Score*. If an input is not in the correct passcode array it was being compared to, zero points will be added to the *Risk Score*. The score is then divided by the length of the pattern.

For example, if the password pattern is set to [2,4,6,1], the brute force attempts would have varying *Risk Score*s, as seen in Figure 5.

[2,8,9,1] = 2 Points (1,0,0,1) = 2/4 = 50%
[1,6,4,2] = 2 Points (0.5,0.5,0.5,0.5) = 2/4 = 50%
[7,8,9,3] = 0 Points (0,0,0,0) = 0/4 = 0%
[1,4,6,2] = 3 Points (0.5,1,1,0.5) = 3/4 = 75%
[2,4,6,1] = 4 Points (1,1,1,1) = 4/4 = 50%

Figure 5. Base pattern *Risk Scores* for varying brute force attempts

4.3. Graphical User Interface

We aimed to replicate current consumer experiences when designing the user interface. We allowed users to set their preferred passcode if it is different from an identified common passcode. Following current standards, a minimum of four nodes must be used to create a valid passcode. The *"Set"* option only becomes available once four nodes are selected during pattern creation. However, during the unlock phase, the *"Unlock"* button is available from the beginning of user interaction. This configuration avoids providing the attacker with information on password credentials or settings. Giving passcode information to an attacker, such as the minimum length of nodes, provides no benefit for device security. For this same reason,

information regarding the *Risk Score* is also not displayed towards the user. After the passcode is set, the user is given a demo on the *Risk Score* meaning. The user may then choose what level of risk acceptance they would like to set their device to for later entry.

4.4. Human Feedback

For testing, we contacted individuals that were familiar with the Android pattern unlock system and recruited them to test out our program through the user interface.

We had 12 participants join our study. The participants completed tasks such as setting a passcode with our precautions in place and participating in mock shoulder surfing to brute force attacks.

4.4.1. Setting the Passcode

The participants were instructed to navigate through the user interface without any influence from the researchers. We reduced interaction so that we were able to collect unbiased information about their experience. The participants set an unlock pattern on a device configured with our interface (Figure 6). If the user attempted to set their passcode to one of the common patterns, they would not be permitted to do so and must try again. We took feedback on their experience after their passcode was successfully set.

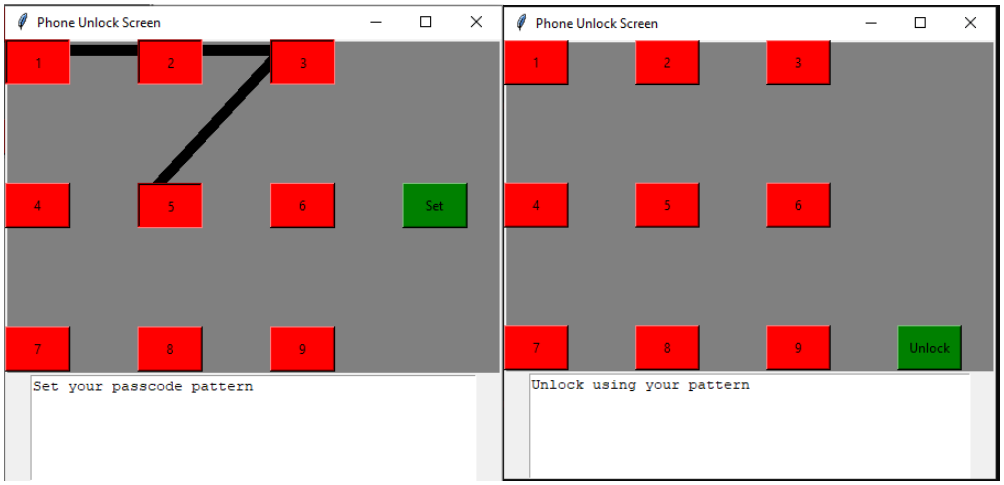

Figure 6. Setting the pattern lock

4.4.2. Simulated Shoulder Surfing Attack

In this stage, we conducted six attack scenarios that each involved two participants. One participant was instructed to sit or stand (based on personal preference) and input their passcode on the device. The other participant walked past the device user to shoulder surf. After three shoulder surfing observations, the attacker was given the device to attempt a brute-force attack. From the attack simulations and participant feedback, we acquired the following results.

5. RESULTS

5.1. Setting the Passcode

Twelve users participated in pattern setting with our interface. On average, it took participants ~ two (1.917) attempts to set an accepted pattern. User satisfaction was measured on a Likert scale of 1 to 5, with 1 being *Low User Satisfaction* and 5 being *High User Satisfaction*. The average user satisfaction was 80% (4/5), as shown in Table 1 below. The high satisfaction shows that the pattern restrictions have minimal impact on usability. The data from this testing can be found in Table 2.

Table 1. User feedback on setting new pattern

User #	Attempt Until Valid Password (/5)	Feedback (/5)
User 1	2	2
User 2	4	3
User 3	1	5
User 4	3	3
User 5	1	5
User 6	2	3
User 7	1	5
User 8	2	5
User 9	2	5
User 10	2	4
User 11	1	3
User 12	2	5
Average	1.916	4

Table 2. Attack success rate with the new imposed algorithm

User	passcode length	Attacker # of tries before phone gets locked	Attacker breach	Feedback
User 1	6	1	No	5
User 2	6	1	No	5
User 3	6	2	No	4
User 4	6	3	No	4
User 5	9	4	No	3.5
User 6	7	2	No	4
User 7	6	0	Yes	2
User 8	6	3	No	3
User 9	8	2	No	4
User 10	7	1	No	5
User 11	5	0	Yes	0
User 12	4	0	Yes	1
Average	6.333	1.583	25%	3.375

5.2. Feedback on the Algorithm

It is crucial to the success of our proposed security method that there is still a suitable level of usability satisfaction from users [13]. We tested the pattern matching algorithm with participants who are all current Android users. After the simulated attack, the shoulder surfer was then given the device and attempted to replicate the observed pattern lock. The majority, 66% (8/12), of the testers agreed that the proposed algorithm would increase the security of the pattern lock and

provide a better recovery model than the current methods on their devices. The participant feedback emphasized that email recovery is the preferred device lockout solution.

5.3. Simulated Attack Success Rates

Our pattern-matching algorithm performs best with pattern locks of length six or higher. We found that shoulder surfing attacks had lower success rates with the implementation of our algorithm. The algorithm has proved its efficiency as it reduces the 65% (13/20) success attack rate from one shoulder surfing observation to 25% (1/4) [5]. In testing, we found a user satisfaction rate of 79% (7.9/10) using patterns lock of length six or higher (Table 2).

User satisfaction for passcodes over the length of six showed a success rate of 79% and 67.5%, which includes passcodes that started from length four.

Eliminating the use of common passcodes received an agreement rate of 80%. Shoulder surfing attacks were successful on passcodes with lengths less than six as they had a 100% success rate. Interestingly, passcodes of six nodes or more only had a 10% success rate.

6. CONCLUSION

Through extensive planning, development, and testing, our algorithm improves the security of any device using a pattern unlock system.

Implementing a system that blocks the use of common patterns, users will be encouraged to create more complex lock patterns that are more difficult for an attacker to brute force. The detection of high-risk pattern unlock attempts helps to keep brute-force attackers from compromising a device.

Finally, the recovery email system helps secure the device by indefinitely blocking interaction from an attacker. There may be limitations to the email method, such as users that do not have another device to access their account on.

Testing showed high user satisfaction overall and supports that our system would increase security without having any noticeable negative impact on usability.

7. LIMITATIONS AND FUTURE WORK

Though we were successful in creating a secure system for increasing pattern unlock security, there are still areas that require improvement. A key component is the *Risk Score* which increases based on the input patterns by the user.

Our interface could improve with more features that allow the user to alter their preferred *Risk Score*. One such addition could be to implement geolocation technology. The *Risk Score* could change based on the location of the device. For example, the score would have more allowance if the phone was in a trusted area, such as the owner’s home.

Human biometrics could be incorporated into a model that recognizes the owner's behaviour, such as touch characteristics.

When setting the pattern, the user could receive better feedback about the strength of their passcode by using tools such as the efficacy meter previously discussed. These improvements could consider features such as similarity to common patterns and pattern length.

We focused on user experience for testing since we could receive direct creative feedback while evaluating our program usability. Expanding the human feedback sample size would provide higher quality insights for future work. Additionally, implementing a control group for comparison of efficacy would give meaningful comparison. Our evaluations could improve with an automated suite that could test our system's effectiveness against automated attacks.

ACKNOWLEDGEMENTS

The authors would like to thank the participants of the study that gave their time and feedback to help verify the efficacy of the proposed model. They also send appreciation to all the academic contacts that contributed suggestions and support to help this paper come to fruition.

REFERENCES

[1] C. W. Munyendo, M. Grant, P. Markert, T. J. Forman, and A. J. Aviv, "Using a Blocklist to Improve the Security of User Selection of Android Patterns," 2021, pp. 37–56.

[2] S. Higashikawa, T. Kosugi, S. Kitajima, and M. Mambo, "Shoulder-Surfing Resistant Authentication Using Pass Pattern of Pattern Lock," *IEICE TRANSACTIONS on Information and Systems*, vol. E101-D, no. 1, pp. 45–52, Jan. 2018.

[3] Y. Ku, L. Hyun Park, S. Shin, and T. Kwon, "Draw It As Shown: Behavioral Pattern Lock for Mobile User Authentication," *IEEE Access*, vol. 7, pp. 69363–69378, 2019, doi: 10.1109/ACCESS.2019.2918647.

[4] L. de Wilde, L. Spreeuwers, and R. Veldhuis, "Exploring How User Routine Affects the Recognition Performance of a Lock Pattern," in *2015 International Conference of the Biometrics Special Interest Group (BIOSIG)*, Sep. 2015, pp. 1–8. doi: 10.1109/BIOSIG.2015.7314603

[5] V. V. Rao and A. S. N. Chakravarthy, "Analysis and bypassing of pattern lock in android smartphone," in *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, Dec. 2016, pp. 1–3. doi: 10.1109/ICCIC.2016.7919555.

[6] L. Bošnjak, J. Sreš, and B. Brumen, "Brute-force and dictionary attack on hashed real-world passwords," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2018, pp. 1161–1166. doi: 10.23919/MIPRO.2018.8400211.

[7] E. Fatima, M. Ashfaq, A. Nazir, M. H. Khan, and M. S. Umar, "A Shoulder Surfing Resistant Technique for Login on Mobile Devices," in *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, Dec. 2018, pp. 1–4. doi: 10.1109/CCAA.2018.8777590.

[8] A. J. Aviv, J. T. Davin, F. Wolf, and R. Kuber, "Towards Baselines for Shoulder Surfing on Mobile Authentication," *Proceedings of the 33rd Annual Computer Security Applications Conference*, pp. 486–498, Dec. 2017, doi: 10.1145/3134600.3134609.

[9] G. Ye *et al.*, "A Video-based Attack for Android Pattern Lock," *ACM Trans. Priv. Secur.*, vol. 21, no. 4, p. 19:1-19:31, Jul. 2018, doi: 10.1145/3230740.

[10] Y. Wang, W. Qiu, Y. Xie, and Y. Zha, "PatternMonitor: a whole pipeline with a much higher level of automation for guessing Android lock pattern based on videos," arXiv:2102.01509 [cs], Feb. 2021, Accessed: Mar. 12, 2022. [Online]. Available: http://arxiv.org/abs/2102.01509

[11] Y. Song, G. Cho, S. Oh, H. Kim, and J. H. Huh, "On the Effectiveness of Pattern Lock Strength Meters: Measuring the Strength of Real World Pattern Locks," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, New York, NY, USA, Apr. 2015, pp. 2343–2352. doi: 10.1145/2702123.2702365.

[12] L. Zhang, Y. Guo, X. Guo, and X. Shao, "Does the layout of the Android unlock pattern affect the security and usability of the password?," Journal of Information Security and Applications, vol. 62, p. 103011, Nov. 2021, doi: 10.1016/j.jisa.2021.103011.

[13] W. Aiken, H. Kim, J. Ryoo, and M. B. Rosson, "An Implementation and Evaluation of Progressive Authentication Using Multiple Level Pattern Locks," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, Aug. 2018, pp. 1–6. doi: 10.1109/PST.2018.8514215.

AUTHORS

Yasir Al-Qaraghuli is currently a student at the University of Guelph in the Masters of Cybersecurity and Threat Intelligence program. He has completed a B.Sc. in Computer Science from the University of Toronto, and Applied Science University in Jordan. He has professional experience working as a Mobile Application Developer which inspires his research focus.

Caroline Hillier is a current student at the University of Guelph enrolled in the Masters of Cybersecurity and Threat Intelligence program. In 2021 she graduated from Trent University with a B.Sc. in Forensic Science and Biology. She has worked on digital security projects which inspires her research objectives of creating secure environments for end users.