Jason Eissayou

11/2/22

CS4960-003

The Android Pattern Lock Side-Channel Attack

Abstract

The Android pattern lock attack proposed by Ye et al. in the paper "A Video-based Attack for Android Pattern Lock"(Ye et al., 2018) is a type of side-channel attack that uses a video recording of the victim swiping in the pattern on their Android phone. Using this video that can easily be taken from a smartphone camera from up to 2.5 meters away, the victim's pattern lock can be deciphered using a mixture of different algorithms. The display on the victim's phone does not need to be visible at all, as all that is needed to be visible to conduct this attack is the victim's finger interacting with the screen. There are five key steps in this algorithm which consist of filming and preprocessing the video, tracking the fingertip location, transforming the filming angle of the video, ranking the candidate patterns, and testing the highest-ranked candidate patterns. Many different factors can also affect the accuracy of this attack. Five of the factors with the largest impacts on the success rate of the attack that are looked into include the impact of filming distance, camera shake, lighting, filming angle estimation, and the grid size of the pattern lock.

I. Introduction

According to research done by the Pew Research Center (2021), there has been a massive growth in the ownership of smartphones over the past ten years. The Pew Research Center shows that in the first study that they did in 2011, only 35% of people in the U.S. owned a smartphone,

while in 2021, 85% of people in the U.S. said they owned a smartphone. Because of this shift in more people buying these smartphones, it can be assumed that more people have started to store sensitive information on these devices. Although smartphones allow people to store data on a small mobile device, keeping more information on these devices greatly increases the threat of cyber criminals stealing this sensitive data.

Cybercriminals can take many different approaches when trying to steal information from companies and people. These approaches can range from launching attacks remotely to stealing physical devices. One significant way that cybercriminals can access and steal data is through cyber reconnaissance. Cyber reconnaissance is described as the phase of a cyberattack "where the goal is to gather as much information about the target as possible" (Brathwaite, 2021). To secure their phones, people can have some type of security lock on the device to ensure they are the only ones who can access the information stored on their phones. To gain access to the phone, cybercriminals can use one of many cyber reconnaissance techniques to help them gain enough information to open the security lock on the phone.

According to the paper "Complexity Metrics and User Strength Perceptions of the Pattern-Lock Graphical Authentication Method", 82.5% of the people surveyed reported that they use the Android OS. Due to this, it is beneficial for criminals to focus on unlocking this Android OS through techniques such as cyber reconnaissance, since a much larger number of people use Android OS.

II. Cyber Reconnaissance

Looking deeper into what cyber reconnaissance is and how it works, Mazurczyk & Caviglione start by presenting three different cyber reconnaissance models in the paper "Cyber

Reconnaissance Techniques" (Mazurczyk & Caviglione, 2021). The three popular cyberattack models presented in this paper include "The Tao of Network Security Monitoring", "Cyber Kill Chain", and "Unified Kill Chain"(Figure 1). Most of these models are described as kill chains because a cyber kill chain is a "series of steps that trace stages of a cyberattack from the early reconnaissance stages to the exfiltration of data" (Hospelhorn 2016). Figure 1 has the names of each of the steps in these models plotted out. According to this figure, in all three of these models of cyberattacks, gathering information is most frequently the first step in initiating a cyberattack.

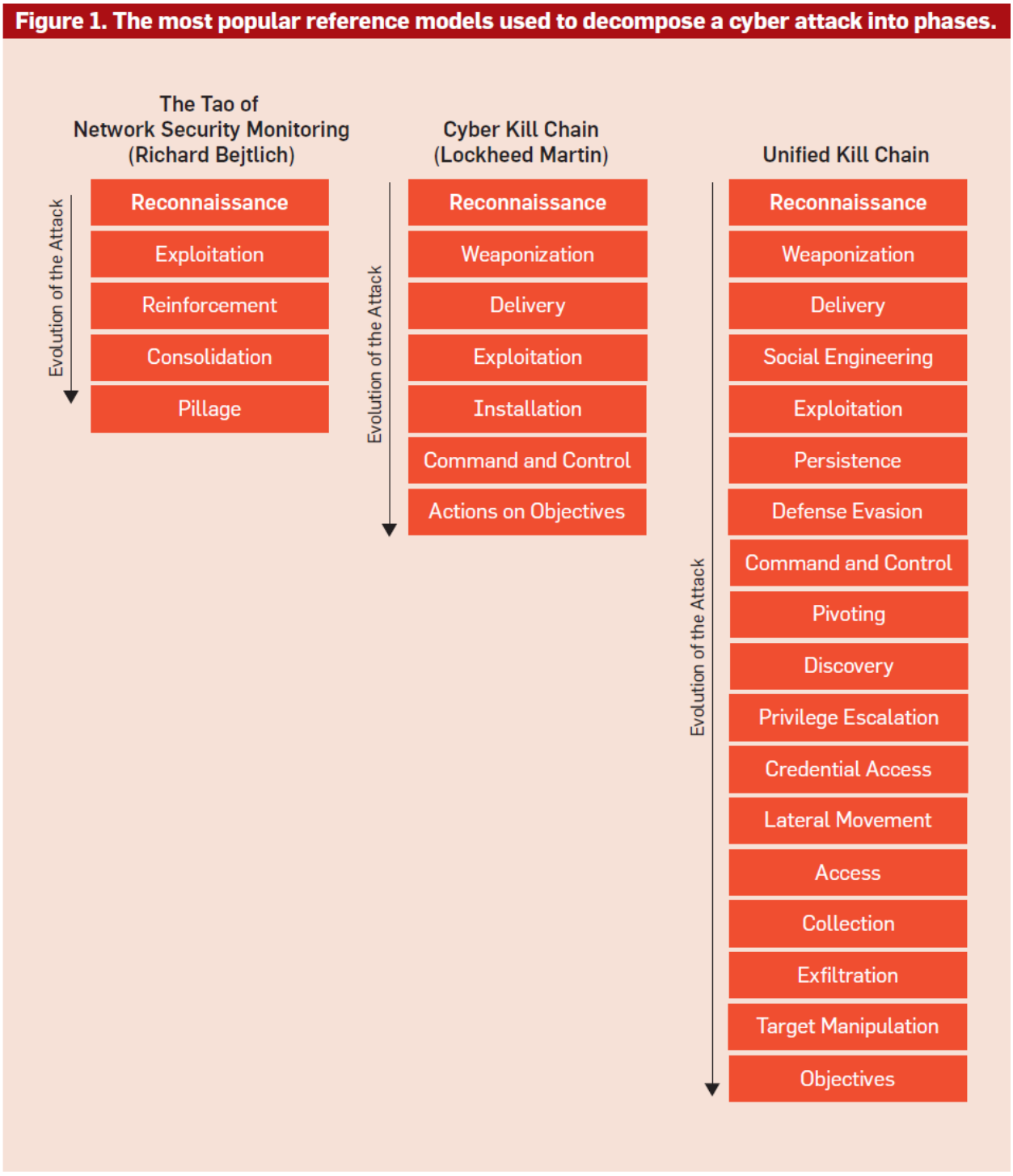


Figure 1. Popular Reference Models of Cyber-Attacks (Mazurczyk & Caviglione, 2021)

There are four main cyber reconnaissance classes as defined by Mazurczyk & Caviglione. The first class defined is called social engineering. This type of cyber reconnaissance is where the attacker interacts with the victim and deceives them to collect information and start the attack. This can include things such as shoulder surfing to collect information or even sending phishing links to the victim. According to Mazurczyk & Caviglione, due to the nature of social engineering, there is a very high degree of interaction between the attacker and the victim. They explain, “reading the computer screen requires to be near the victim, thus potentially having a physical interaction…” (Mazurczyk & Caviglione, 2021, 88).

The second class that is defined is internet intelligence. Internet intelligence shows that the attacker doesn't have to interact with the victim at all, like in social engineering. Instead, the attacker looks at publicly made available information about the victim to gain the data they need for the attack, which calls for almost no degree of interaction between the attacker and the victim.

The third class of cyber reconnaissance is network information gathering. This class consists of scanning the victim's network and gathering information this way. For example, the attacker can connect to the victim’s network and use different network sniffing tools to look at the traffic going through the network and steal information this way.

The final class is called side-channel attacks. A side-channel attack is where a victim leaks information unknowingly, and in turn, the attacker uses this leaked information to gather the information they need to formulate an attack. An example of this type of attack is explained in the article “Acoustic Side-Channel Attacks on Printers”. As authors Backes et al. (2010) have figured out, using the noise that a Dot-Maxtrix printer makes, it is possible to formulate the

English text that a printer is processing. Looking at all of these different types of techniques, attackers have many options when it comes to conducting cyber reconnaissance.

III. "A Video-based Attack for Android Pattern Lock" Introduction

The pattern lock attack proposed by Ye et al. in the paper "A Video-based Attack for Android Pattern Lock" relies on using a video of the user swiping in their pattern lock to unlock their phone. Due to this, it is a cyber reconnaissance attack that uses information that the user doesn't know they're leaking, which makes this attack a type of side-channel attack. The overall premise of this attack and how it works is that it is a video-based attack using a video of the victim unlocking their Android phone. Using this video, which does not even need to capture what is displayed on the victim's screen, the attacker can use the victim's fingertip motions and the multiple algorithms described in the paper to reconstruct the victim's lock pattern. The attack presented in this paper is broken up into five major parts. The first major part of this attack is filming the video and preprocessing it to ensure that the two areas of interest are marked. The video is preprocessed using an algorithm to make sure that it is cut only to the segment where the user swipes in their pattern, making this attack much easier when taking long videos. Two areas of interest which consist of the victim's fingertip and the edge of the device are then marked. After all of the preprocessing, the video is then run through a computer vision algorithm. This algorithm tracks the fingertip of the victim to map out a rough pattern to unlock the phone. Since the camera is usually at an angle though, the pattern mapped out by this algorithm is still off and would not be viable. To fix this, the third stage of this attack involves using different algorithms to transform this pattern from the camera's perspective to the victim's perspective using the edge of the device so the software can work with the video. In the second to last stage, the software

proposed by this paper looks at all the different possible combinations and ranks them. It then chooses no more than five possible combinations for the pattern lock. Finally, in the last step, the attacker tries these possible combinations given by the algorithm on the victim's phone.

IV. Pre-Processing The Video

Looking into the first step of this attack, the authors must take the video and preprocess this video to make it usable. To do this, a heuristic proposed by the authors called the "Unlocking Process Identification Heuristic" is used to preprocess this video. The heuristic defined in Figure 2 is pseudocode for the algorithm created by the authors so it does not dive deep into each of the functions and variables. Looking at this algorithm in Figure 2, it is split into two parts: the input and the output. The inputs of this algorithm include two different variables, the video footage that the attacker has recorded (IV) and the pause threshold that the attacker has chosen (frameCount). This pause threshold is used to make a distinction between two different on-screen operations. According to research done by Ye et al., "all our participants paused at least 1.5s before or after pattern drawing due to delay of the user or the device" (19:6), and because of this, the pause threshold is set to 1.5 seconds. Since the code requires the number of frames and not the number of seconds, it must also be translated into frames. Most cameras usually shoot at 30 or 60 frames per second and this translates to 45 or 90 frames per 1.5 seconds, respectively. Therefore, depending on the type of camera the attacker uses, this frameCount value will usually be either 45 or 90.

Moving on to the output of the algorithm, the algorithm starts off in lines 1 and 2 by defining two of the main variables. In line 1, each frame of the video footage is stored in the

“frames[]” array. After this, the length of the array is also found and stored in the “LEN” variable.

After defining these two variables in lines 1 and 2, the main “for” loop of the algorithm is started on line 3. The “for” loop sets i=1 and loops through every 1.5 seconds of the entire footage, as stated by LEN - frameCount.

Inside the “for” loop, the first argument on line 4 uses a function “hasFingertipChanged()” with the inputs i and frameCount. What this function does is check if the fingertip location has changed in the 1.5-second increments. This function then returns True or False and this value is assigned to the variable sL. If the fingertip location has changed and the hasFingertipChanged function returns True, this means that the user has not taken the observed 1.5-second pause, and the heuristic goes to line 17 to end the “if” statement and restarts the “for” loop.

If instead the hasFingertipChanged function returns false, that means that the victim has taken at least the 1.5-second pause that was observed to be taken before or after swiping in the pattern. The variable sL is then set to False and because of this, we enter the “if” statement on Line 5. The heuristic then defines a variable sNo which corresponds to the beginning of when the victim takes this 1.5-second pause to set the starting frame of the victim swiping in their password.

Line 7 then starts another “for” loop to figure out where the ending frame is going to be. There are three different exit conditions in this “for” loop. The first two exits in this “for” loop include the two different “if” statements in lines 8-10 and lines 11-13. If any of these “if” statements are entered, that means an ending frame has been found. If nothing has been found, the loop is broken with no endpoint.

The first “if” statement on lines 8-10 checks if there are two consecutive swiping or zooming motions and if there are, the frame is stored into the variable eNo and the loop is broken. The reason for setting the frames after a consecutive swipe or zoom is that the authors found out through research that “identical on-screen activities often follow closely (after pattern drawing)”(Ye et al., 2018, 19:6).

If there are no consecutive swiping motions, the algorithm moves on to the second “if” statement. This “if” statement uses the same hasFingertipChanged function to check if there is another 1.5-second pause from the victim. If the algorithm detects this 1.5-second pause, it sets the frame where the pause was detected into the variable eNo and breaks the loop.

After one of these “if” statements break the loop, the main “for” loop is finished and the algorithm finally moves on to line 19, where it defines the start and end of the targeted video segment. In this line, a function called getTargetVideo() is used that takes in three inputs. These inputs include the frames[] array which was the frames of the entire video, sNo which was the starting frame of the targeting segment, and eNo which was the ending frame of the targeted segment. Using these three arguments, the algorithm outputs the start and end of the targeted video segment.

ALGORITHM 1: Unlocking Process Identification Heuristic

Input:
IV: Video footage
$frameCount$: Pause threshold before or after unlocking
Output:
<start,end>: Start and end of the unlocking video segment

```
1:  frames[] ← getVideoFrames(IV)
2:  LEN ← getFramesLen(frames[])
3:  for i = 1 : LEN − frameCount do
4:      sL ← hasFingertipChanged(frames[i : i + frameCount])
5:      if !sL then
6:          sNo = i + frameCount
7:          for j = sNo : LEN do
8:              if checkLoop(frames[j : LEN]) then
9:                  eNo = i
10:                 break;
11:             else if !hasFingertipChanged(frames[j : j + frameCount]) then
12:                 eNo = i
13:                 break;
14:             end if
15:         end for
16:         break;
17:     end if
18: end for
19: < start, end >← getTargetVideo(frames[], sNo, eNo)
```

Figure 2. Unlocking Process Identification Heuristic (Ye et al., 2018)

V. Tracking The Fingertip

After the preprocessing of the video is done and the targeted video segment is located, the next step in this attack is to use an algorithm to track the fingertip locations of the victim. The authors use an algorithm created by Zdenek Kalal et al. called Tracking-Learning-Detection (TLD). How this algorithm works is that for each object TLD is tracking, it assigns a confidence value to the object between 0 and 1. When this confidence value is greater than a threshold set by the attacker, TLD tracks the object it was set to track.

One major problem that the authors discovered with this tracking algorithm is that it gives the location of the tracked object relative to the pixel at the top left of the screen. This is

seen as a problem because when a video is taken using a handheld phone, it is not completely steady since humans tend to move a little bit when trying to hold something still. As some scientists state, "…the mental effort required to hold your hand still causes the muscles in your hand to contract a bit at regular intervals and therefore experience a tremor, even under normal circumstances" (Bolz, 2010). To solve this problem, the authors used a method where instead of just tracking the fingertip of the victim, the system tracks two different areas of the recording. The first area tracked is the fingertip, while the second area is the edge of the mobile device. By doing this, the authors can find out the coordinates of the fingertip relative to the edge of the device. This allows the algorithm to filter out the error from the camera shake since instead of giving coordinates for the fingertip position relative to a fixed point, it will give the coordinates relative to the edge of the phone. This ensures that the fingertip is compared to an accurate dynamic coordinate if the camera is shaky.

VI. Filming Angle Transformation

Once the fingertip tracking is complete, there is still a major chance the coordinates might still differ largely from the actual lock pattern. This is because the video of the victim's phone is usually taken at an angle. It is very difficult for an attacker to record the mobile device directly face-on without causing suspicion and because of this, recording the victim's phone at an angle is a situation that needs to be taken into account. The authors have devised a solution to this problem using a detection algorithm called Line Segment Detector (LSD) (Grompone et al., 2010). Using this algorithm, the authors are able to detect the long edge of the phone and compare this angled edge to a vertical line. The angle between these two lines is then taken and used to transform the trajectory of the fingertip from the camera's perspective to the user's

perspective. The transformation is done according to the formula shown in Figure 3. S', as seen in the formula, is the original coordinate of the tracked fingertip. This original coordinate is then transformed to S by multiplying the original point by the variable T, which is defined as the transformation matrix seen in Figure 3. Since the video of the victim unlocking their phone is taken on a handheld device, angles could change due to movement from the victim or the attacker. Due to the possibility of such movement, the algorithm must make this transformation for each separate frame.

$$S = TS' \quad , \quad T = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix},$$

Figure 3. Transformation Matrix (Ye et al., 2018)

VII. Map out Candidate Patterns

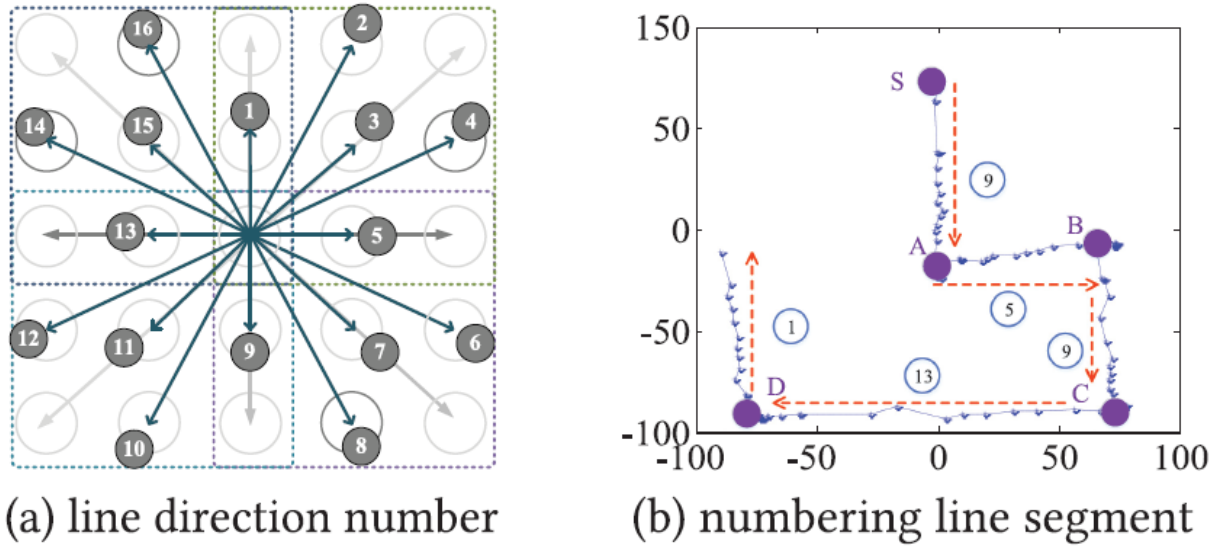
After transforming all of the data, the last step of this attack is to map out the candidate patterns. To find the correct pattern, the system starts with a list of many patterns and removes the ones that do not match the criteria that were given to the system.

Before going deeper into how these patterns are mapped, the definition of a pattern as described by the authors of this attack needs to be defined. A pattern is described as "...a collection of line segments…"(Ye et al., 2018, 19:10) where these line segments have two different variables, length (L) and direction (D). To put this into variable terms, P={L, D} where L and D are both arrays with the length and direction information for each line segment.

Now that it is understood how the authors have defined a pattern, one can now look into how these line segments are actually identified. To figure out what each line segment is, the turning points of the finger must be identified. The authors use a linear fitting method developed by Kutner et al. (2004) to figure out these turning points. A deep dive into this algorithm isn't required but what this algorithm does is that it finds the turning points of the finger so the authors will be able to break each swipe into a separate line segment. One problem of this linear fitting method that had to be overcome was that if two lines overlapped, the method would believe it was only one line. To fix this, the authors had to use the temporal information of when the swipe was made to separate these line segments.

After figuring out all the turning points, the length and distance information for each line segment must be extracted. Starting with the length of each line, one main problem needs to be tackled. Since mobile devices come in varying sizes, an approach that works on devices of any size must be used. To achieve this, the authors devised a solution where the system uses the shortest line segment found and normalizes this as the length of a single line segment. After finding the length of each line segment, the second attribute of the pattern must be found, which is the direction of the line segment. In Figure 4(a), the authors have created a layout and numbering for all the different directions. Each number corresponds to a different type of swipe in varying directions. To be able to map out these swipes, the authors pre-define a slope for each type of numbered swipe which is shown in Figure 5. Now that the slope and direction are defined for each type of swipe, Figure 4(b) is an example of how all of these swipes can be put together and numbered using all the information defined in Figures 4(a) and 5. The pattern in Figure 4(b) starts off at the “S” dot and goes down to the “A” dot. Due to the direction and slope of this line, referring back to Figures 4(a) and 5 shows that this swipe should be matched with the number 9.

The rest of the swipes are then numbered using the same method used to number the “S” to “A” swipe.



Figures 4a and 4b. Line Directions and Example Trajectory (Ye et al., 2018)

Direction No.	1	2	3	4	5	6	7	8
slope (L → R)	$+\infty$	2	1	$\frac{1}{2}$	0	$-\frac{1}{2}$	-1	-2
Direction No.	9	10	11	12	13	14	15	16
slope (R → L)	$-\infty$	2	1	$\frac{1}{2}$	0	$-\frac{1}{2}$	-1	-2

Figure 5. Direction to Slope Mapping (Ye et al., 2018)

To start off identifying these patterns, the system compares the number of turning points to the number of line segments of the possible patterns. If the pattern has more line segments compared to the turning points, it is automatically rejected. Following this, the length and directions of the line segments are looked at. If one of the possible mappings does not correspond with the length and directions of the lines, it is rejected. After these rounds of

rejections, the system ranks all of the remaining patterns using a heuristic. The heuristic that the authors chose has one main assumption: A pattern starting from a dot on the left side of the screen is chosen over a pattern starting from a dot towards the rightmost side. This is because, from research done by (Løge, 2015 & Uellenbeck et al., 2013), it is found that the left dot is usually selected to start the pattern. When the same dot is a starting point for two patterns, the pattern with the longer line lengths will be ranked higher.

When all these patterns are finally ranked, the system picks the top five patterns and presents them to the attacker. The reason for choosing the top five patterns is explained by the author as "the Android system by default allows more than five fail attempts, and an attacker can test all the five patterns in a short period of time" (Ye et al., 2018, 19:12). Presented with these five patterns, the attacker now has all the information needed to unlock the phone.

VIII. Impact of Filming Distance, Camera Shake, Lighting, and Filming Angle Estimation

Since this attack relies on filming a video of someone unlocking their phone, this attack is prone to different factors that can affect the consistency of the algorithm. The four main factors discussed in this paper are the impact of filming distance, the camera shake, the lighting, and finally, the filming angle estimation.

Starting with the impact of the filming distance, the authors tested the accuracy of the algorithm when recorded with a smartphone camera from different distances. To test the accuracy from different distances, it is explained that "we used all the 120 collected patterns and we varied the filming distance from 1 to 3.5m" (Ye et al., 2018, 19:18). The authors were able to plot all of this information on a graph which can be seen in Figure 6. Through this graph, it can be seen that when the algorithm has five attempts to solve the pattern lock, the accuracy is almost

100% up until the subject is around 2.5 meters away. After 2.5 meters, the accuracy drops significantly for a variety of reasons. The one main reason for this drop that the authors state is that the quality of the video taken by the smartphone drops significantly when the subject is more than 2.5 meters away. This in turn makes it very hard for the algorithm to crack the pattern lock since at this distance, "the video quality makes it difficult for the TLD algorithm to successfully track objects across video frames" (Ye et al., 2018, 19:18). Although this can be a major issue for the attacker, the authors explain that attackers can instead opt for a high-end camera if a high-quality recording from a far distance is required.

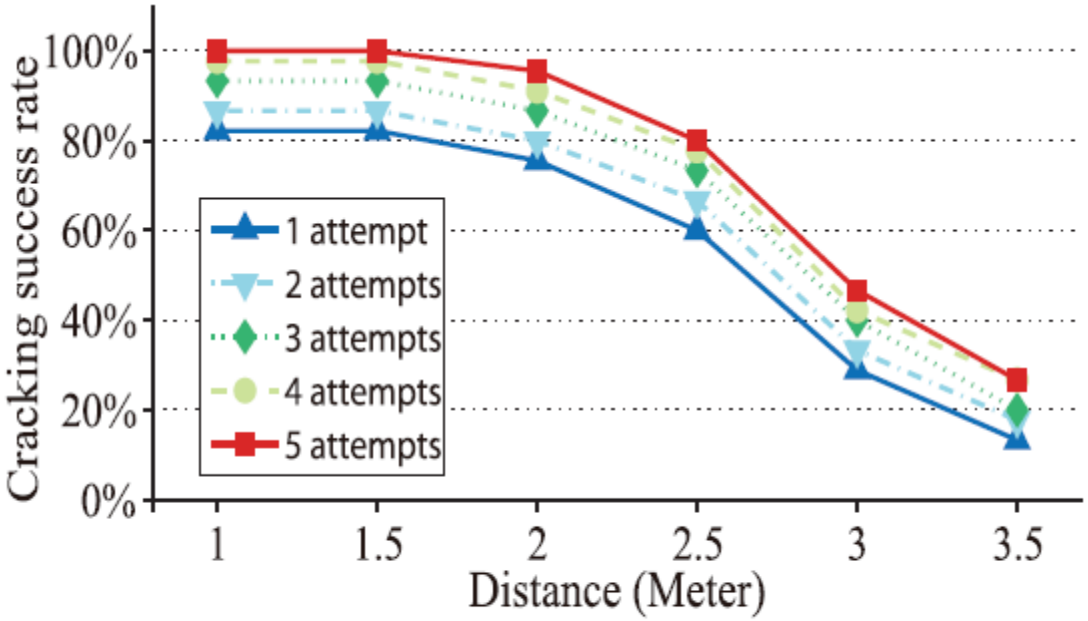


Figure 6. Filming Distance Impact (Ye et al., 2018)

Camera shake was also a significant factor when looking into the success rate of the algorithm. To set up the experiment of how detrimental camera shake was to the algorithm's success, the authors recorded videos using three different settings. These settings included using a tripod, hand-held, and hand-held but with constant movements of around 2cm to simulate

shaking. Again, the authors plotted all their findings in a graph shown in Figure 7. Through this figure, the author's camera shake calibration is shown to work very well, as the hand-held recording yields virtually the same accuracy as the tripod in almost every test. The very shaky camera did have an accuracy degradation of around 15%-20%, but as the authors state: "In reality, an attacker would avoid drastic camera shake by firmly holding the video recording device" (Ye et al., 2018, 19:19). This, in turn, lead the algorithm to have the same accuracy handheld as it would when using a tripod.

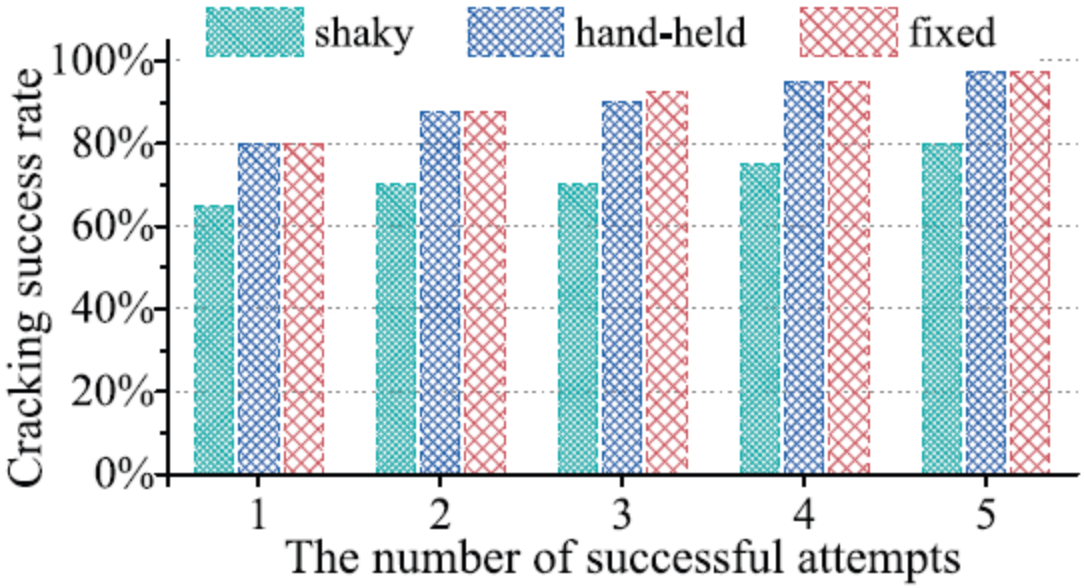


Figure 7. Camera Shake Impact (Ye et al., 2018)

Moving on, the authors also tested the impact of lighting on the success rate of the algorithm. Lighting was tested in setting ranging from 55 lux to 9500 Lux. When the lighting was at its lowest (55 Lux), the accuracy of this algorithm was still around 80%. The accuracy maxes out to 100% at 150 Lux and continues to be 100% all the way to the max Lux that was tested (9500 Lux). To put this all into perspective, the article "Lux, Lumens and Watts: Our

Guide" explains the amount of Lux in typical areas that people usually visit. Through this article, it is shown that a typical University classroom has around 250 Lux, which is well over the Lux needed for the accuracy to reach 100% for this algorithm. A family living room has around 50 Lux though, which lowers the accuracy of this algorithm to around 80%. Anything lower than this amount of lighting will drastically reduce the success rate of this algorithm.

The final factor discussed by the authors that can affect the success rate of this algorithm is the camera filming angle estimation. Looking back at the algorithm, the algorithm must transform the movement of the finger seen from the camera to the movement from the user's perspective. The authors explain that their filming angle estimation algorithm gives very accurate results and there is no reason for there to be an error in the estimation, but for data purposes, they still calculate how an error in this estimation can affect the success rate. It is explained that even with an error of 5 degrees, the algorithm still is highly accurate. It is when the error exceeds 10 degrees that the success rate of the attack takes a huge hit. As the author explains though, "we did not find the estimation error to be an issue in our experiments" (Ye et al., 2018, 19:20), so the error in the filming angle estimation isn't a large issue since there is a very slim chance it will reach this 10 degrees of error.

IX. Pattern Grid Size

Another attribute of the pattern lock that can affect the algorithm is the size of the pattern lock. Although the default pattern grid of the Android OS is 3x3, different tools are downloadable to increase the size of the pattern grid. Many different applications can be downloaded to do this, but the most popular is CyanLockScreen which is one such application that allows users to increase the grid size from the default 3x3 size to bigger grids such as 4x4,

5x5, and 6x6 (Arrows, 2022). When researching the attack on larger grids, Ye et al. discovered that the accuracy of the attack did, in fact, drop when dealing with larger grids. Figure 8 shows that the accuracy of the attack drops from around 95% on a 4×4 grid to 87% on a 6×6 grid on complex grids and from 85% on a 4 × 4 grid to 75% on a 6 × 6 grid on simple grids (Ye et al., 2018, 19:21).

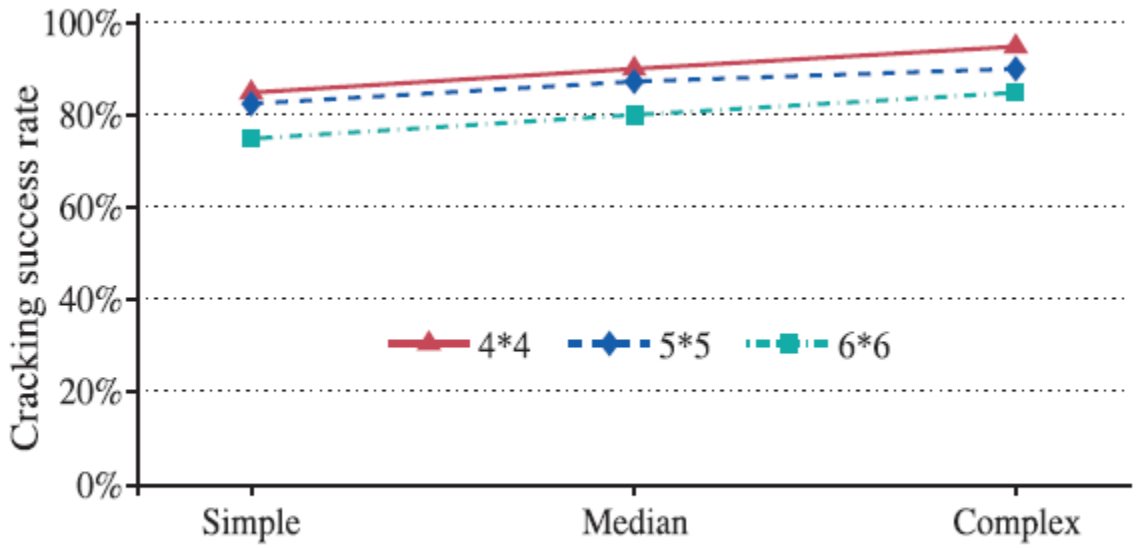


Figure 8. The success Rate For Different Types of Grids (Ye et al., 2018)

Through these numbers, it can be seen that a larger pattern grid was, in fact, more secure than smaller pattern grids. Although larger pattern grids are more secure, looking at Figure 8, the lowest success rate was still around 75% which is still very successful when dealing with the large number of pattern possibilities that a 6x6 grid allows one to create.

The majority of Android phones continue to keep a 3x3 lock screen though. This is because downloading these types of apps that allow you to change the size of the pattern grid requires root access to the android phone (Arrows, 2022). According to a study done by Tencent in 2014, around 27.44% of Android users have enabled root access on their phones (Lucic,

2014). This shows that about 72% of Android users have no way to increase their pattern grid size, thus allowing a high success rate of the attack.

X. Conclusion

A large variety of factors must work together to allow this side-channel attack to succeed. This attack had five main parts: video preprocessing, tracking the fingertip, transforming the filming angle, ranking all possible combinations and picking the top five, and finally, using the top five pattern combinations on the device lock. Also, as explained, there were a large number of factors that could affect the success rate of the attack. The filming distance, camera shake, lighting, and filming angle estimation all had a significant role in the attack's success. A user may also have an Android phone with a grid bigger than 3x3, which also affects the success rate. Although many factors can lower the success rate of this attack, it still has a very high chance of succeeding due to all of the safeguards the authors have implemented into this algorithm. Because of this, this video-based attack for an Android pattern lock is a viable side-channel attack that can be used to be able to access personal information stored in a password-protected device.

Personal Statement

Throughout researching and writing this paper, one major thing I learned was what cyber reconnaissance is and how it works. I wasn't aware of the four different types of cyber reconnaissance classes such as social engineering, internet intelligence, network information gathering, and side channel. I was able to learn the underlying structure of each of these attacks and how these attacks worked in real-world applications.

Not only was I able to learn about these classes of cyber reconnaissance and how they worked, but I also learned how dangerously easy it was to conduct cyber reconnaissance and steal information from unknowing victims. This really helped me think about what type of information I may be leaking and how I could stop this to minimize the risk of a cyberattack being conducted on me.

Through studying the Android Pattern Lock Side-Channel Attack, I found out how easy it was to leak information by just doing things such as unlocking your phone in public. Things such as typing your password on your computer, or any password for that matter, can easily be recorded in public without you knowing. I was also amazed at how much data can be stolen from a victim with very little information. One small thing like a phone password can lead to attackers accessing very important things such as bank accounts. I usually give out my iPhone passcode to friends but after doing research on this topic, I have decided to change my password and not share it with anybody.

Works Cited

Andriotis, P., Tryfonas, T., & Oikonomou, G. (2014). Complexity metrics and user strength perceptions of the pattern-lock graphical authentication method. *Lecture Notes in Computer Science*, 115–126. https://doi.org/10.1007/978-3-319-07620-1_11

Arrows, K. (2022, August 18). *How to increase pattern lock size on Android*. Appuals.com. Retrieved October 25, 2022, from https://appuals.com/increase-pattern-lock-size-android/

Aviv, A. J., Gibson, K., Mossop, E., Blaze, M., & Smith, J. M. (2010). Smudge attacks on smartphone touch screens. *Proceedings of the 4th USENIX Conference on Offensive Technologies*, 1-10. https://dl.acm.org/doi/10.5555/1925004.1925009

Backes, M., Dürmuth, M., Gerling, S., Pinkal, M., & Sporleder, C. (2010, August 1). Acoustic side-channel attacks on printers. *Proceedings of the 19th USENIX Conference on Security*, 1-20. https://dl.acm.org/doi/10.5555/1929820.1929847

Bolz, B. (2010, March 29). *Why do our hands have a nervous tremor?* A Moment of Science - Indiana Public Media. Retrieved October 25, 2022, from https://indianapublicmedia.org/amomentofscience/hands-nervous-tremor.php#:~:text=However%2C%20scientists%20believe%20that%20the,tremor%2C%20even%20under%20normal%20circumstances

Brathwaite, S. (2021, February 23). *Active vs passive cybersecurity reconnaissance in information security*. SecurityMadeSimple. Retrieved October 25, 2022, from https://www.securitymadesimple.org/cybersecurity-blog/active-vs-passive-cyber-reconnaissance-in-information-security

Hospelhorn, S. (2022, February 24). *What is the cyber kill chain and how to use it effectively*. Varonis. Retrieved November 1, 2022, from https://www.varonis.com/blog/cyber-kill-chain

Kalal, Z., Mikolajczyk, K., & Matas, J. (2011). Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence, 6(1)*, 1-14. http://vision.stanford.edu/teaching/cs231b_spring1415/papers/Kalal-PAMI.pdf

Kutner, M. H., Nachtsheim, C., & Neter, J. (2004). *Applied Linear Regression Models* (5th ed.). McGraw-Hill/Irwin.

Løge, M. D. (2015, July 13). *"Tell me who you are and I will tell you your unlock pattern"*. NTNU Open. Retrieved October 25, 2022, from https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2380967

Lucic, K. (2014, November 13). *Over 27.44% users root their phone(s) in order to remove built-in apps, are you one of them?* Android Headlines. Retrieved October 25, 2022, from https://www.androidheadlines.com/2014/11/50-users-root-phones-order-remove-built-apps-one.html#:~:text=According%20to%20Tencent's%20study%2C%2027.44,many%20people%20actually%20use%20Android.

Mazurczyk, W., & Caviglione, L. (2021). Cyber Reconnaissance Techniques. *Communications of the ACM*, *64*(3), 86–95. https://doi.org/10.1145/3418293

Pew Research Center. (2022, October 7). *Mobile fact sheet*. Pew Research Center: Internet, Science & Tech. Retrieved October 25, 2022, from https://www.pewresearch.org/internet/fact-sheet/mobile/

Read our ultimate guide to lux vs lumens vs Watts for lighting installations: Warehouse & Factory Lighting. Green Business Light. (n.d.). Retrieved October 25, 2022, from https://greenbusinesslight.com/resources/lighting-lux-lumens-watts/

Uellenbeck, S., Dürmuth, M., Wolf, C., & Holz, T. (2013, November 1). Quantifying the security of graphical passwords. *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, 161-172. https://dl.acm.org/doi/10.1145/2508859.2516700

Ye, G., Tang, Z., Fang, D., Chen, X., Wolff, W., Aviv, A. J., & Wang, Z. (2018). A video-based attack for Android Pattern Lock. *ACM Transactions on Privacy and Security*, *21*(4), 1–31. https://doi.org/10.1145/3230740