A Systematic Deconstruction of Human-Centric Privacy & Security Threats on Mobile Phones

Habiba Farzand[a,*], Melvin Abraham[a], Stephen Brewster[a], Mohamed Khamis[a] and Karola Marky[b]

[a]University of Glasgow (UK), [b]Ruhr University Bochum (Germany)

ARTICLE HISTORY
Compiled May 28, 2024

ABSTRACT
Mobile phones are most likely the subject of targeted attacks, such as software exploits. The resources needed to carry out such attacks are becoming increasingly available and, hence, easily executable, putting users' privacy at risk. We conducted a systematic literature analysis to understand the relationship between resources and attack feasibility and present a categorisation of social engineering and side-channel attacks on mobile phones focusing on the resources attackers require. Our proposed categorisation levels facilitate an in-depth understanding of how mobile phone attacks can be executed using different combinations of partly simple resources. The analysis reveals that attackers exploit users' unawareness about the surrounding context of the device's usage to invade users' privacy and that discrete protection mechanisms are insufficient to provide all-inclusive protection. The proposed categorisation assists in building novel solutions for safeguarding users' security and privacy from diverse attacks by carefully considering the potential misuse of resources. We conclude by outlining future research directions highlighting the urgent need for a holistic user defense.

KEYWORDS
social engineering attacks, side channel attacks, human-centred attacks, mobile devices

1. Introduction

Social engineering and side-channel attacks are two commonly explored attack categories (Aldawood & Skinner, 2019; Devi & Majumder, 2021b; Ivaturi & Janczewski, 2011; Joy Persial, Prabhu, & Shanmugalakshmi, 2011; Krombholz, Hobel, Huber, & Weippl, 2015; Song et al., 2016). While any technological device could be a target of such attacks, handheld mobile devices are an attractive target for attackers, due to the rich data they can collector data entered by users which could include personal information (Hölzl, Roland, & Mayrhofer, 2017), health-related data (Hernandez, McDuff, & Picard, 2015), voice interactions (Amazon, 2022; Apple, 2022; Google, 2022) or emotional states (Grünerbl et al., 2014; LiKamWa, Liu, Lane, & Zhong, 2013).

Methods to attack mobile devices are manifold (Clarke & Furnell, 2005; La Polla, Martinelli, & Sgandurra, 2012). Yet, attackers mostly require specific resources to attack mobile users, such as a malicious app installed on the user's device or external

*Corresponding Author Habiba Farzand (h.farzand.1@research.gla.ac.uk) requisites, like a video recording device, to capture the user's input. Prior work has proposed several taxonomies around social engineering or side-channel attacks (Aldawood & Skinner, 2019; Ivaturi & Janczewski, 2011; Johnson, 2021). Yet, the ease of attack execution based on required resources has not received in-depth attention from the research community. For this, we investigate the following research question:

RQ1: What resources are required to perform privacy and security attacks on mobile devices, such as social engineering and side channel attacks?

This paper addresses this gap by investigating the resources attackers need for successful attacks. To achieve this, we performed a systematic literature review on social engineering and side-channel attacks by selecting the top 10 publication venues in "Human Computer Interaction" and the top 10 publication venues in "Computers Security & Cryptography" according to the Google Scholar ranking system. Additionally, we checked papers published at SOUPS (i.e., the Symposium on Usable Security and Privacy, co-located with the USENIX Security Symposium) since it covers the intersection research of HCI and security. We extracted attack requirements from the systematic literature review, including resources (e.g., specific hardware, software or knowledge) from the resulting papers from the systematic literature review.

The resources required to perform a specific attack are quite versatile, making it challenging to compare different attacks to assess their likelihood or severity. Without categorisation, it is difficult to understand which attack is easier to perform than another. This directs us towards our second research question:

RQ2: How can privacy and security attacks be methodically categorized to reflect the ease of execution?

Using the extracted list of resources, we developed a categorisation that establishes a hierarchy of security and privacy attacks on mobile devices based on their required resources, indicating their ease of execution. Our proposed categorisation is four-layered: (1) Novice Attacks, (2) Intermediate Attacks, (3) Proficient Attacks, and (4) Expert Attacks.

Our investigation shows that possible attacks on mobile devices have become quite ubiquitous. They are no longer limited to the physical location of users. Furthermore, the barriers for laypeople without specific knowledge to becoming attackers are low due to recent advances in attack tools. Based on that, we can conclude that one does not even have to be a so-called *"script kiddie"* anymore because human capabilities (e.g., observation by looking at a device) and manual tools (e.g., paper and pen) are already sufficient to invade the privacy of mobile users. Our proposed categorisation assists researchers and practitioners in classifying (existing and future) attacks based on attack requirements. This knowledge helps estimate the scalability and frequency of privacy attacks and provides new perspectives in designing novel and comprehensive privacy-preserving mechanisms. Our categorisation further enhances the development of social engineering and side channel attack mitigation mechanisms and measures.

Research Contribution. The contribution of this paper is manifold:

1) **In-depth literature review:** We present an in-depth literature review about resources that attackers need to carry out attacks on handheld mobile devices.
2) **Categorisation based on requirements:** We systematically investigate the resources and organize them into a four-layered categorisation of (1) Novice At-

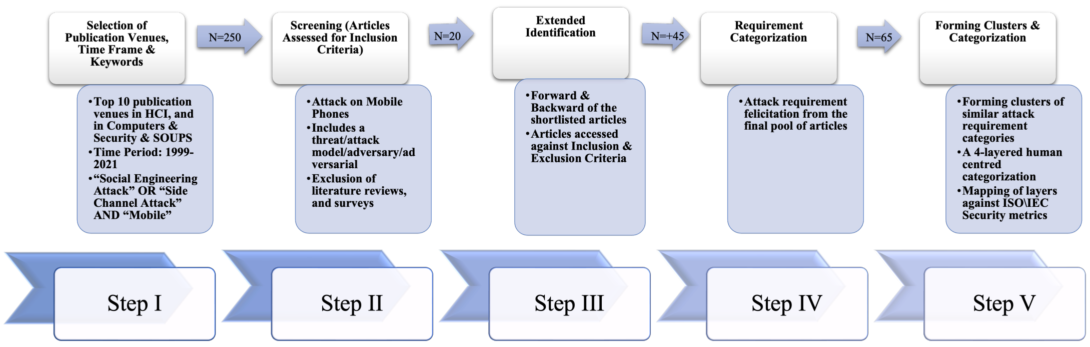

Selection of Publication Venues, Time Frame & Keywords
N=250
Screening (Articles Assessed for Inclusion Criteria)
N=20
Extended Identification
N=+45
Requirement Categorization
N=65
Forming Clusters & Categorization
•Top 10 publication venues in HCI, and in Computers & Security & SOUPS
•Time Period: 1999-2021
•"Social Engineering Attack" OR "Side Channel Attack" AND "Mobile"
•Attack on Mobile Phones
•Includes a threat/attack model/adversary/adversarial
•Exclusion of literature reviews, and surveys
•Forward & Backward of the shortlisted articles
•Articles accessed against Inclusion & Exclusion Criteria
•Attack requirement felicitation from the final pool of articles
•Forming clusters of similar attack requirement categories
•A 4-layered human centred categorization
•Mapping of layers against ISO\IEC Security metrics
Step I
Step II
Step III
Step IV
Step V

Figure 1. The figure shows the step-wise systematic literature review methodology we followed to develop the categorisation of Social Engineering and Side-Channel Attacks on Mobile Phones.

tacks, (2) Intermediate Attacks, (3) Proficient Attacks, and (4) Expert Attacks. Our categorisation provides an in-depth overview of attack resources.

3) **Highlighting the ease of attacks:** Our work shows that any individual can easily become an attacker, for example, by using human capabilities and manual tools. More sophisticated attacks can become more accessible for individuals due to the easily available resources such as malware. Finally, our research highlights the urgent need for a better defence of users on a more holistic level rather than placing an additional burden on users to opt for individual countermeasures for individual attacks or overloading them with the need to be aware of attacks 24-7.

2. Related Work

The first group of taxonomies focuses on one particular *category* of social engineering attacks. Among them, the taxonomy from Heartfield and Loukas specifically considers semantic social engineering attacks (Heartfield & Loukas, 2015). Semantic attacks are a category of social engineering attacks that perform an attack by manipulating object characteristics, such as system applications, with the purpose of deceiving as opposed to directly attacking the user. Heartfield and Loukas propose a baseline for classifying semantic attacks by breaking them down into their components and surveying the applicable defences. However, related work has shown that other categories of social engineering attacks can be carried out without interfering with the user-computer interface, such as thermal attacks (Alotaibi, Williamson, & Khamis, 2023). Other work focuses on hardware-based side-channel attacks and analyses the types, mitigation methods, targets, techniques, and methods (Johnson, 2021).

The second group of prior work consists of surveys focusing on one particular *type* of social engineering attack, such as phishing (S. Gupta, Singhal, & Kapoor, 2016). In this context, Gupta et al. (S. Gupta et al., 2016) discuss various methods to perform phishing attacks, their prevention, detection, and their role in the daily lives of people (S. Gupta et al., 2016). Gupta et al. reported that phishing is typically carried out on email spoofing or instant messaging and targets users with little or no knowledge of social engineering attacks or internet security. Gupta et al. further discuss various types of phishing attacks and prevention techniques. Prior work also presented surveys on side-channel attacks in the Internet of Things (IoT) and discussed several significant areas for research and improvement in security (Devi & Majumder, 2021a) while other researchers have also investigated side-channel attacks on critical infrastructures and relevant systems (Tsalis, Vasilellis, Mentzelioti, & Apostolopoulos, 2019). Recent work on security attacks on graphical passwords has explored various attacks for graphical passwords and their countermeasures through literature investigation (Por, Ng, Chen, Yang, & Ku, 2024).

While all these taxonomies and categorisations deliver valuable insights, they either consider one particular category of social engineering attacks or one specific attack type. This paper bridges this gap by presenting the first comprehensive evaluation of social engineering and side-channel attacks considering the resource-oriented nature of attacks from an attacker's perspective. In doing so, different attack types can be compared to each other and evaluated for their ubiquity, frequency, and feasibility, which is important for designing adequate and comprehensive countermeasures.

3. Methodology

This section describes the steps of our systematic literature review on human-centred social engineering and side channel attacks on handheld mobile devices. The adopted methodology is illustrated in Figure 1.

Step 1: Key Words & Search Space: First, we identified keywords iteratively through discussions among three experts from the field. Our search query reflects our research focus on side-channel and social engineering attacks on mobile devices: (*social engineering attack** OR *side channel attack**) AND (*mobile** OR *mobile device** OR *mobile phone** OR *phone** OR *smartphone** OR *personal device** OR *handheld device**). We carefully tested the keywords the experts discussed to overcome the keyword selection bias and made a pre-search to ensure our keywords were not too limited in the search space.

As search space, we selected the top 10 publication venues in "Human-Computer Interaction" and the top 10 in "Computers Security & Cryptography" according to the Google Scholar ranking system (date accessed: May 06, 2021) and set the time of publication from 1999–May 2021 as 1999 is the year when one of the most influential human-centred security papers was published (Adams & Sasse, 1999).We also checked papers published at SOUPS since this is a top venue for usable security and privacy research (Distler et al., 2021).

Step 2: Screening: The search results from Step 1 were then manually inspected for the location of the keywords within the full text. We further screened the papers, based on our inclusion criteria of having a threat, adversary or adversarial model. A threat model explains the vulnerabilities of a system (Ghasempouri et al., 2021) and details how an attack is performed against a target and which resources are needed by the attacker (Rescorla & Korver, 2003) from the perspective of an attacker or a defender (Liu, Zang, & Yu, 2005; Myagmar, Lee, & Yurcik, 2005). Threat models differ from adversary models because adversary models refer to goals, assumptions, and capabilities (Shi, Niu, Jakobsson, & Chow, 2010) representing a general approach to executing attacks. However, as noted by other researchers (Do, Martini, & Choo, 2019), threat and adversary models have been used interchangeably in the literature. To provide maximum coverage of relevant papers, we additionally included papers that provided an adversary model, an adversarial model, an attack model, or an attack overview. Each paper had to provide an attack overview, including a threat model, so that we can extract a complete list of resources the attacker needs without making any assumptions.

Step 3: Backward & Forward Search: For each paper identified in Step 2, we performed a backward and forward search using the same keywords to include any relevant papers published elsewhere other than the selected publication venues. We re-applied the exclusion and inclusion criteria to the papers resulting from Step 3. The complete list of included papers included in the categorisation can be found in Appendix A.

Exclusion Criteria: Papers that included the keywords in references, paper classification, or author's biography were removed from the analysis. We excluded papers that focused on devices other than mobile phones. Further, we excluded papers that were not peer-reviewed, such as papers on ArXiv, bachelor/master's theses, and doctoral theses. For the doctoral theses, we checked the list of references for the same set of keywords and exclusion and inclusion criteria. A paper was excluded if the model description was incomplete and required subjective interpretation. We only included papers written in English. Lastly, we excluded papers with a model that just relied on coercing the victim, e.g., by using physical violence (B. Chang et al., 2018), because our research focuses on the resources and skills of attackers rather than coercion.

Step 4: Extraction of Resources: For each paper, one researcher extracted a list of requirements by copying the information given in the paper. Another researcher verified the resulting list. Next, we followed an inductive categorisation approach to cluster the requirements into groups until categorisation was no longer meaningful (Mayring et al., 2004). This resulted in seven clusters. Two researchers validated the clusters. We refer to the clusters of resources as *requirements*. The coding sheet for requirements can be found in Appendix B.

Step 5: Building the Categorisation: We used the seven clusters of requirements from the previous step to describe each attack identified in the papers. During this analysis, four levels of requirements that build our categorisation emerged: (1) Novice, (2) Intermediate, (3) Proficient, and (4) Expert.

To estimate the resource complexity of an attack in each category, we considered the *security levels* defined by the ISO/IEC Security metrics (62443, 2022; Alliance, 2020; Committee & (TC65WG10), 2016). The ISO/IEC Security metrics are international standards that address cybersecurity used in many research papers such as (Mayrhofer & Sigg, 2021). The *security levels* in particular describe the *measure of confidence that the System Under Consideration, Zone, or Conduit is free from vulnerabilities and functions in an intended manner*" (Alliance, 2020, p.8). More specifically, the ISO/IEC Security metrics describe the level of protection from the system view, considering different types of attackers based on the resources needed to attack a system. These levels are quite generic yet provide a way to protect a system. By mapping the resulting categories of attack resources, we show how easy or difficult an attacker can attack a mobile device. The ISO/IEC 62443 security levels (62443, 2022; Alliance, 2020;

Committee & (TC65WG10), 2016) are as follows[1]:

SL0 "No special requirement or protection required"
SL1 "Protection against unintentional or accidental misuse"
SL2 "Protection against intentional misuse by simple means with few resources, general skills and low motivation"
SL3 "Protection against intentional misuse by sophisticated means with moderate resources, (IACS-specific) knowledge and moderate motivation"
SL4 "Protection against intentional misuse using sophisticated means with extensive resources, (IACS-specific) knowledge and high motivation"

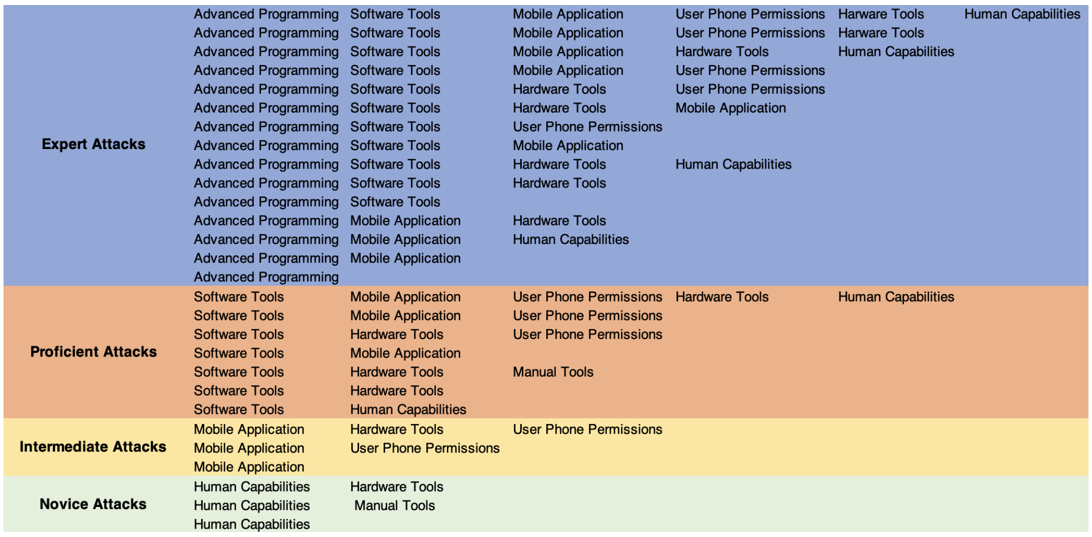
Limitations: Like most literature reviews, our work has several limitations. First, our literature review was conducted in May 2021. Papers published after this time are not considered. We selected "social engineering attacks" and "side channel attacks" as search keywords based on an expert discussion and keywords from relevant papers to focus our search. However, there might be further papers published on these topics that do not include our selected keywords in the full text. Papers that did not match the screening criteria were excluded from the analysis. While this might have shrunk the space of outcome, it produced a final list of papers focused on the criteria mentioned above. Papers that provided fuzzy information about attacks left too much room for subjective interpretation, which would have threatened the validity and reproducibility of our work. Lastly, some of the publication venues did not offer a search function, such as the USENIX Security Symposium. For this, we used Google Scholar to elicit relevant papers. Some papers may have been dropped due to the limitations of Google Scholar as the search engine.

4. Requirements of Human Centred Attacks on Mobile Phones

In this section, we detail the requirement categories that we extracted from the literature. From the final set of papers, we extracted seven categories of requirements. We detail them below with explanations and examples that an attacker may utilize to perform social engineering or side-channel attacks.

1) Software Tools: Refers to benign programs that make use of sophisticated algorithms but are not specifically designed for malicious use. Examples include a remote server, software that implements an n-gram Markov Model, or software that collects fine-grained accelerometer data.
2) Mobile Phone App: A specific app that needs to be installed on the victim's device that is specifically designed for malicious use. Examples include spyware and phishing apps.
3) Advanced Programming: Advanced programming expertise from specialized fields of programming. Examples include knowledge of implementing and executing deep learning, or image processing algorithms.
4) User Phone Permissions: Access to specific sensors and resources that are guided by permissions on the victim's device is required to execute the attack, such as access to WiFi.
5) Hardware Tools: External electronic hardware tools are required in the attack setup. Examples include charging cables and wireless routers.

[1]Please note, that we only consider the attack perspective in terms of requirements and not the protection of the attacked system.



	Advanced Programming	Software Tools	Mobile Application	User Phone Permissions	Harware Tools	Human Capabilities
	Advanced Programming	Software Tools	Mobile Application	User Phone Permissions	Harware Tools	
	Advanced Programming	Software Tools	Mobile Application	Hardware Tools	Human Capabilities	
	Advanced Programming	Software Tools	Mobile Application	User Phone Permissions		
	Advanced Programming	Software Tools	Hardware Tools	User Phone Permissions		
	Advanced Programming	Software Tools	Hardware Tools	Mobile Application		
	Advanced Programming	Software Tools	User Phone Permissions			
Expert Attacks	Advanced Programming	Software Tools	Mobile Application			
	Advanced Programming	Software Tools	Hardware Tools	Human Capabilities		
	Advanced Programming	Software Tools	Hardware Tools			
	Advanced Programming	Software Tools				
	Advanced Programming	Mobile Application	Hardware Tools			
	Advanced Programming	Mobile Application	Human Capabilities			
	Advanced Programming	Mobile Application				
	Advanced Programming					
	Software Tools	Mobile Application	User Phone Permissions	Hardware Tools	Human Capabilities	
	Software Tools	Mobile Application	User Phone Permissions			
	Software Tools	Hardware Tools	User Phone Permissions			
Proficient Attacks	Software Tools	Mobile Application				
	Software Tools	Hardware Tools	Manual Tools			
	Software Tools	Hardware Tools				
	Software Tools	Human Capabilities				
	Mobile Application	Hardware Tools	User Phone Permissions			
Intermediate Attacks	Mobile Application	User Phone Permissions				
	Mobile Application					
	Human Capabilities	Hardware Tools				
Novice Attacks	Human Capabilities	Manual Tools				
	Human Capabilities					

Figure 2. The figure shows the categorisation of Human Centred Social Engineering and Side Channel Attacks on Mobile Phones from the Perspective of an Attacker. We developed this categorization based on the resources extracted from the papers resulting from the systematic literature review.

6) Human Capabilities: Resources that fall within the physical and personal abilities of humans, such as physical access to the device, close proximity in distance, knowledge about the victim, and target observation.
7) Manual Tools: Refers to non-electronic/non-powered devices or tools. Examples include pens and pencils.
A visual representation of the categorisation can be found in Appendix 3.

5. Glossary of Attacks

We propose four categories: (1) Novice, (2) Advanced Beginner, (3) Proficient and (4) Expert (see also Figure 3). When detailing each layer, we also map it to the ISO/IEC Security metrics (62443, 2022; Committee & (TC65WG10), 2016) and present options to counter specific attacks. The following sections detail each of the categories and respective subcategories.

6. Level 1: Novice Attacks

All attacks at this level exclusively rely on human capabilities, manual tools, and basic hardware tools. Human capabilities, such as making observations through human sight, do not require the attacker to acquire special expertise in using specific equipment since the requirements needed to perform the attack are within the capabilities of a human. Manual tools like pens and pencils are readily available and accessible. Basic hardware tools, such as specific cameras, are easy to buy and use. It does not require training or a special setup to use. Given this, the attacks with such requirements can be labelled as "novice attacks". Considering ISO-IEC Security standards (Committee & (TC65WG10), 2016), this attack category corresponds to **SL0** and **SL1** because no special requirements are needed. The following clusters of requirement categories fall under Novice Attacks.

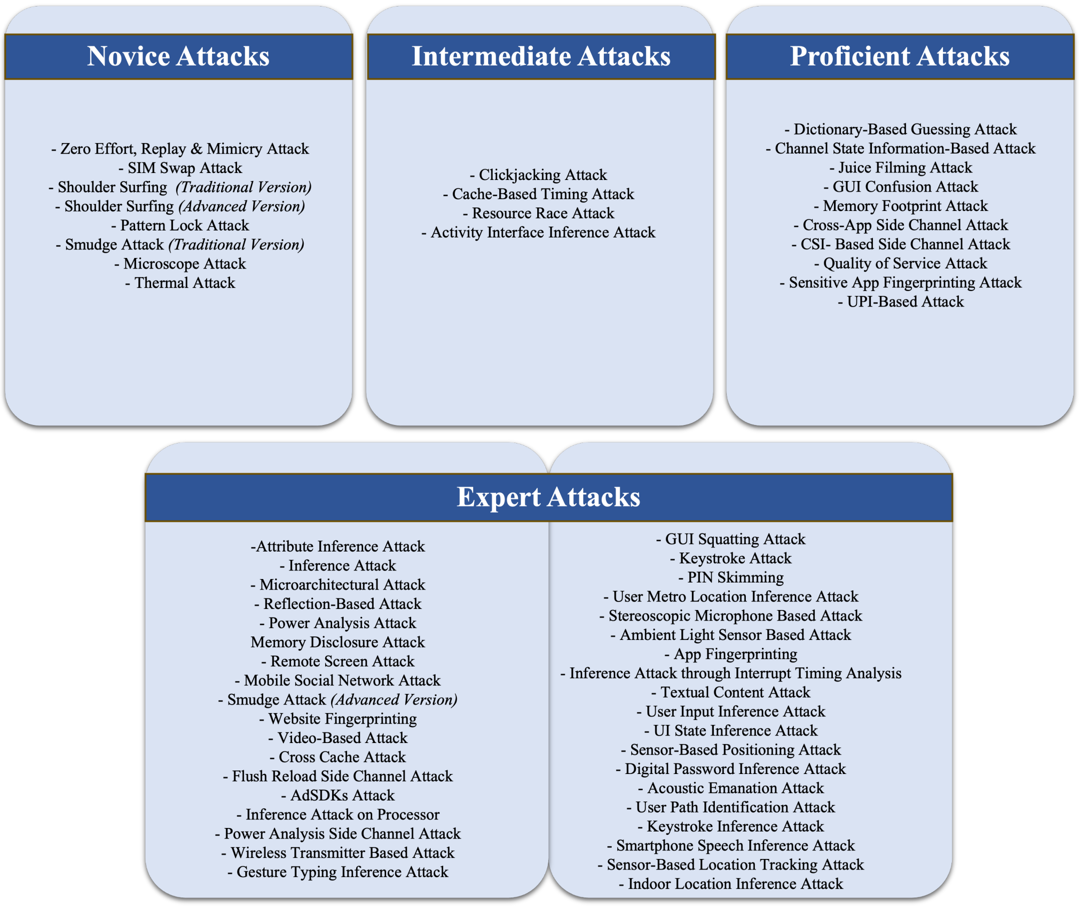

Novice Attacks
- Zero Effort, Replay & Mimicry Attack
- SIM Swap Attack
- Shoulder Surfing *(Traditional Version)*
- Shoulder Surfing *(Advanced Version)*
- Pattern Lock Attack
- Smudge Attack *(Traditional Version)*
- Microscope Attack
- Thermal Attack
Intermediate Attacks
- Clickjacking Attack
- Cache-Based Timing Attack
- Resource Race Attack
- Activity Interface Inference Attack
Proficient Attacks
- Dictionary-Based Guessing Attack
- Channel State Information-Based Attack
- Juice Filming Attack
- GUI Confusion Attack
- Memory Footprint Attack
- Cross-App Side Channel Attack
- CSI- Based Side Channel Attack
- Quality of Service Attack
- Sensitive App Fingerprinting Attack
- UPI-Based Attack
Expert Attacks
-Attribute Inference Attack
- Inference Attack
- Microarchitectural Attack
- Reflection-Based Attack
- Power Analysis Attack
Memory Disclosure Attack
- Remote Screen Attack
- Mobile Social Network Attack
- Smudge Attack *(Advanced Version)*
- Website Fingerprinting
- Video-Based Attack
- Cross Cache Attack
- Flush Reload Side Channel Attack
- AdSDKs Attack
- Inference Attack on Processor
- Power Analysis Side Channel Attack
- Wireless Transmitter Based Attack
- Gesture Typing Inference Attack
- GUI Squatting Attack
- Keystroke Attack
- PIN Skimming
- User Metro Location Inference Attack
- Stereoscopic Microphone Based Attack
- Ambient Light Sensor Based Attack
- App Fingerprinting
- Inference Attack through Interrupt Timing Analysis
- Textual Content Attack
- User Input Inference Attack
- UI State Inference Attack
- Sensor-Based Positioning Attack
- Digital Password Inference Attack
- Acoustic Emanation Attack
- User Path Identification Attack
- Keystroke Inference Attack
- Smartphone Speech Inference Attack
- Sensor-Based Location Tracking Attack
- Indoor Location Inference Attack

Figure 3. The figure shows the categorisation of attacks in four levels from an attacker's perspective. The expertise required to perform an attacker increases as we progress in categorisation levels.

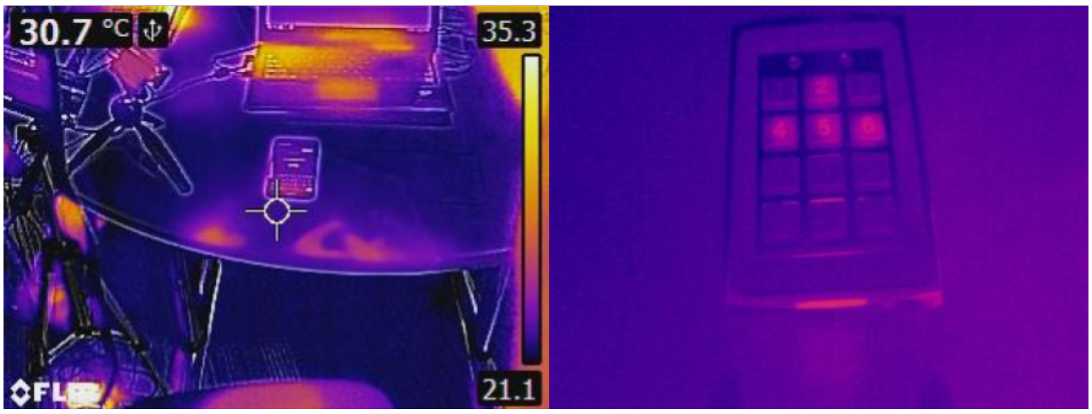


Figure 4. The figure shows a thermal image of smartphone authentication captured using a thermal camera, i.e. a Flir camera. The authentication information can be easily observed by observing the heat traces. This is a typical scenario of Thermal Attacks that belongs to the category of Novice Attacks.

6.1. Human Capabilities

By utilizing the human capability of making observations through sight or using personal information about the victim, an attacker can uncover the victim's mobile device content, replay the gained knowledge to gain unauthorized access to the user's device or transfer SIM contract details to another SIM card number. Attacks such as zero effort, replay, and mimicry (Wu, He, Chen, & Du, 2019) can only be efficiently performed using human capabilities. No external resources are required. Similarly, a SIM swap attack (Lee, Kaiser, Mayer, & Narayanan, 2020) can be performed by convincing the carrier to update the SIM card linked to the victim's phone. For this, the attacker only needs to know the victim's name and phone number and have access to auto-refill interfaces. Shoulder surfing (traditional) (Dunphy, Heiner, & Asokan, 2010; Mitchell, Wang, & Reiher, 2015) - also referred to as one of the out-of-device threats Farzand, Marky, and Khamis (2024), is another attack that can be performed by making observations of the victim's device screen. The attacker can uncover confidential and private information by observing the screen as the victim interacts with the device. Though shoulder surfing is mostly reported on smartphones (Eiband, Khamis, Von Zezschwitz, Hussmann, & Alt, 2017; Farzand, Marky, & Khamis, 2022), its evidence is found in interaction with multiple tech devices such as Virtual Reality, and researchers have proposed numerous mechanisms to combat it (Huestegge & Pimenidis, 2014; Wang et al., 2023).

6.2. Human Capabilities & Manual Tools

By utilizing the human capability of making observations through sight with some manual tools, an attacker can uncover the victim's mobile device content with which the victim interacts. Adding manual tools to human capabilities contributes to the attack's success. An example of such an attack is shoulder surfing (advanced). The traditional shoulder surfing attack is advanced by adding manual tools for note-taking, such as a pen or pencil. The attacker aims to capture the victim's device authentication pattern (such as Pass sketches (Yu et al., 2017)) by observing their input.

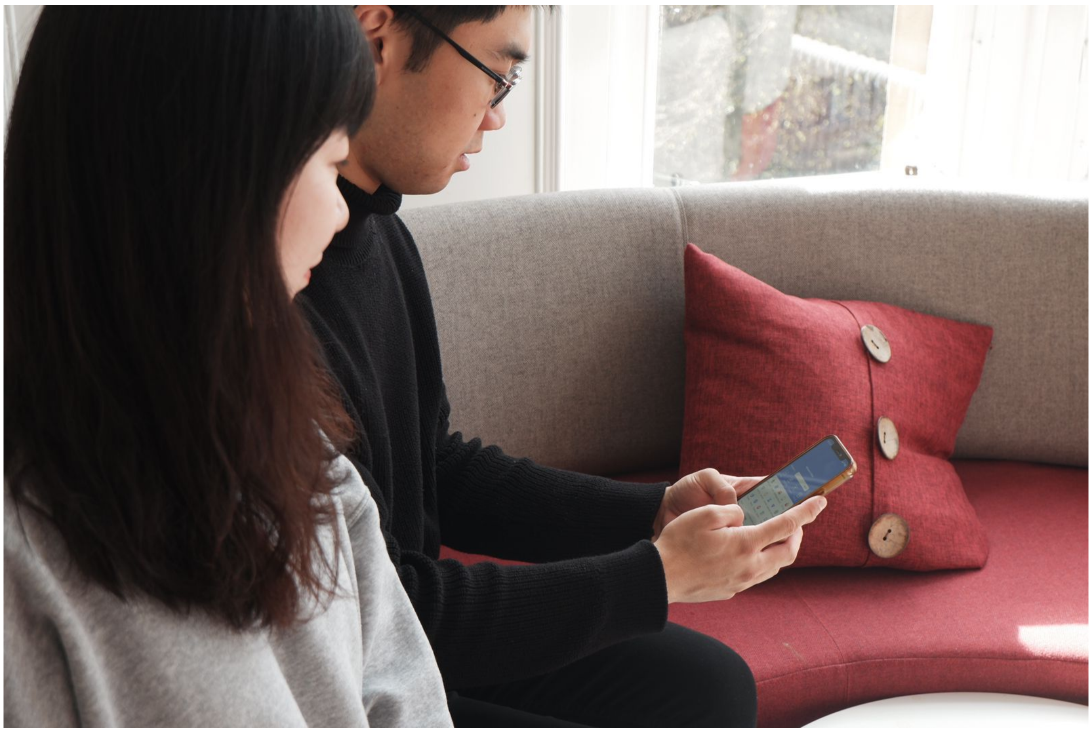


Figure 5. The figure showcases one of the common scenarios of shoulder surfing - a type of Novice Attack - in the daily lives of users where the bystander uses direct observation to make observations of the screen and is able to retrieve personal information about the user.

6.3. Human Capabilities & Hardware Tools

Utilising human capabilities and some basic hardware tools, such as a recording device, can assist an attacker in performing several attacks, especially authentication-based attacks. Examples of such attacks include pattern lock attacks (Ku, Park, Shin, & Kwon, 2019), smudge attacks (traditional) (Andriotis, Tryfonas, Oikonomou, & Yildiz, 2013), microscope attacks (Andriotis et al., 2013), and thermal attacks (Andriotis et al., 2013). In the case of a pattern lock attack, the attacker is close to the victim and uses a recording device (for example, a smartphone's camera) to record authentication steps, including input patterns and gestures. Then, the attacker gets physical access to the victim's device to unlock it. Similarly, to perform smudge attacks, an attacker inspects smudge residue left by the victim's fingers on the device to reconstruct credentials. For this, the attackers need physical access to the device and some hardware tools, such as a compact camera to capture the smudges on the screen and a hard light source to get the edged shadows. By manually inspecting the image, the attacker can reconstruct the credential.

Authentication information a user enters can be uncovered by utilizing a microscope attack. For this, the attacker has to be in close proximity to the target device and then use a high-definition camera to capture an image of authentication while it is being performed. The attacker can then utilize a microscopic device, such as a USB microscope with 400x magnification, to deduce the entered information. Like a microscope attack, authentication information can be uncovered using a thermal camera. When using a thermal camera, the attacker needs a thermal camera and should be close to the user or have physical access to the device. The attacker can then capture a thermal image of the screen right after the authentication information has been entered, and then, by manual inspection, the attacker can unveil the entered information.

7. Level 2: Intermediate Attacks

Attacks belonging to this category require limited effort for practical implementation, namely, a combination of mobile apps, hardware, and manual tools, as well as human capabilities. Hardware tools are easily accessible and offered at cheap prices by various vendors. Hence, access to hardware tools is no longer a difficult task to accomplish. Manual tools and human capabilities also come at a minimal price and effort. Hence, the only arrangement the attacker needs to make is to prepare a mobile application. With a combination of a mobile app with hardware tools, human capabilities, and manual tools, various attacks could be performed. Considering ISO-IEC standards (62443, 2022; Alliance, 2020; Committee & (TC65WG10), 2016), this attack corresponds to **SL2**. The following lists the combinations of requirements that attackers can utilize to compromise a victim user's privacy and security.

7.1. Mobile Apps

A mobile app can be used in a malicious way to learn sensitive information about users. A classic example is clickjacking (Possemato, Lanzi, Chung, Lee, & Fratantonio, 2018), where by clicking on an overlay created over the victim app, an attacker tricks the user to, e.g., grant permission. The overlay is created by a malicious app, which is opaque in the foreground. While thinking about performing a legitimate action, the user only visually sees the victim app while running in the background while interacting with the malicious app. This way, only through developing a mobile app and tricking the user into installing it on his device the attacker learns the sensitive information. Indistinguisly, a cache-based timing attack (N. Zhang, Sun, Shands, Lou, & Hou, 2016) also requires only a mobile app. The mobile app is used to extract sensitive information by exploiting the cache contention between the normal world and the secure world.

7.2. Mobile Apps & User Phone Permissions

A mobile app with few permissions can perform a resource race attack (Cai et al., 2020). The race between mobile apps to access resources can be exploited to steal sensitive information. For example, a malicious app can steal sensitive information from a legitimate app that captures it, e.g., photos.

7.3. Mobile Apps, Hardware Tools & Phone Permissions

With the addition of hardware tools to a mobile app with access to some target phone permissions, an attacker can perform an activity interface inference attack (Yang, Zhi, Wei, Yu, & Ma, 2019). The goal of this attack is to uncover the user's app activity. The attacker makes use of the shared memory side-channel information. The attacker develops a malicious app with access to the Internet as well as the device storage and installs it on the victim's device. The app continuously collects information about the foreground application processes and uses it in a training data phase to build an activity signature database. Then, in the attack phase, the malicious app sends the collected characteristic data to a server for calculations. It then uses the signature database to uncover the activity.

8. Level 3: Proficient Attacks

The attacks falling in this category require increased effort, resources, and capabilities. The development of a mobile app along with software tools is necessary to perform the attacks. Phone permissions are not difficult to access. Most attacks require Internet permission only, which is marked as a ”PROTECTION_NORMAL” by Android (for Developers, 2022; H. Security, 2022). Moreover, it has been observed that users do not pay attention to the permissions being granted to apps (Felt et al., 2012). Hence, it has become easy for an app to get access to permissions. The complex part of this category is that the attacker must have software skills and must know how to develop malicious apps. There are multiple ways to manipulate the user to install malicious apps without realizing the malicious intent behind them (cf. (Goel & Jain, 2018)). Considering ISO-IEC standards, this attack category corresponds to **SL3**.

8.1. Software Tools & Human Capabilities

Pattern-based authentication systems could be attacked and leaked while using software tools along with human capabilities. Dictionary-based pattern guessing attacks (Cho et al., 2017) are a blueprint of such attack categories. A Pattern Dictionary-Based attack requires physical access to the target device and trying the most probable unlock patterns to access the phone. The goal is to unlock the phone in less than 20 trials because mobile OSes lock the phone and require their users to log in through their mobile OS account after 20 failed attempts. To unlock the phone in 20 attempts or less, the attacker uses a probabilistic password model, such as the n-gram Markov model (Ma, Yang, Luo, & Li, 2014). This model could be trained using real-world data of unlock patterns. While this attack is easy to carry out, this attack methodology cannot be applied to every mobile OS.

8.2. Software & Hardware Tools

Adding hardware tools to software tools can assist an attacker in performing channel state information-based attacks (J. Zhang et al., 2019). In such an attack scenario, the adversary aims to access sensitive information entered on the victim’s device. To make this attack possible, at least one or two wireless devices, such as a wireless router, a laptop, or a smartphone, must be placed within 0.5-5 meters distance from the target device in a static setting. The wireless devices must support ICMP protocol and communicate CSI readings.

8.3. Software, Hardware & Manual Tools

A new kind of charging attack called juice filming attacks (Jiang, Meng, Wang, Su, & Li, 2017; Meng, Lee, Murali, & Krishnan, 2015) can be performed using software tools with the assistance of hardware and manual tools. In a typical juice filming attack, the attacker records user inputs, e.g., by a VGA/USB interface that is connected to the smartphone via a malicious charger (Meng et al., 2015). The VGA/USB interface is concealed in the user's environment. No app needs to be installed on the target smartphone, and no user permission is required.

8.4. Software Tools & Mobile Apps

GUI-based (Bianchi et al., 2015; Fernandes et al., 2014), memory footprint(Jana & Shmatikov, 2012) and cross-app side-channel attacks (X. Zhang, Wang, Bai, Zhang, & Wang, 2018) can be executed when software tools are combined with a mobile app.

With the goal of performing a GUI-based attack, which is sometimes also referred to as Pixel Perfect Phishing Attack (Fernandes et al., 2014), the attacker makes use of a malicious app installed on the victim's phone. The app has the ability to merge multiple attack vectors, such as UI-intercepting draw-overs (Felt & Wagner, 2011; Niemietz & Schwenk, 2012), toast messages (Niemietz & Schwenk, 2012), non-UI-intercepting draw-overs (Felt & Wagner, 2011; Luo, Jin, Ananthanarayanan, & Du, 2012; Niemietz & Schwenk, 2012) and enhancing techniques, such as app repackaging (Hanna et al., 2012; Zhou, Zhang, & Jiang, 2013; Zhou, Zhou, Jiang, & Ning, 2012), accessing the proc file system (Jana & Shmatikov, 2012) to perform attacks. The malicious app presents itself as a benign app, e.g., as a utility. When the malicious app is launched, it monitors other apps on the victim's phone and waits until a target app is launched. The malicious app can be a look-alike version of the target app that discloses any information entered into a remote server.

Similarly, in a memory footprint attack, again, a mobile app with some software tools is sufficient to bypass user privacy. In a memory footprint attack, two processes (victim and attack process) running in parallel on the same host can learn the secrets of web browser processes by tracking the changes in the app's memory footprint. As a first step, the attacker profiles the target program and creates an attack signature database through a malicious app. Next, the attack process measures the memory footprint of the victim process. The attacker can download the activity signature database or send the attack memory footprints to a remote server for matching.

Using the same set of resources as required to perform a memory footprint attack, an attacker is capable of performing cross-app side-channel attacks. This attack exploits side-channel information leakage on the OS level. A malicious app is installed on the victim's device, runs in the background and collects traces for each event of interest. For each time series, the difference between two consecutive points is noted, and then SAX transformation and BOP construction are performed. Next, the attacker converts BOP into LibSVM and uses LibSVM to perform classification. RBF kernel could be used for SVM classification and a probability model to perform cross-validation.

8.5. Software Tools, Hardware Tools & User Phone Permissions

When user phone permissions are complemented by software tools with the addition of some hardware tools, attacks like keystroke inference can be easily implemented. CSI-based keystroke inference attack (M. Li et al., 2016) is a classic example of such attack resources. Users who use public WiFi can be victims of CSI-based keystroke inference attacks. To perform this attack, the attacker requires the victim's device to be connected to public WiFi. This is commonly seen in public spaces such as restaurants, shopping malls, airports and alike. The WiFi hotspot usually has an application layer security (HTTPS) that helps in gaining user trust that the connection is safe. Upon getting the device connected, the WiFi hotspot collects CSI from the victim's device through ICMP protocol/ Further, through a directional antenna, the noise in CSI is eliminated. Li (M. Li et al., 2016) proposed an algorithm for keystroke recognition. They adopted a low pass filter to remove high-frequency noises and Principal Component Analysis to reduce the dimensionality of the feature vectors. They also proposed a context-oriented CSI-collected method to recognize the PIN input.

8.6. Software Tools, Mobile Apps & User Phone Permissions

A combination of software tools and mobile applications with access to permissions can assist in performing fingerprinting and performance degradation attacks such as Quality of Service (QoS) attack (Inci, Eisenbarth, & Sunar, 2017) and sensitive apps fingerprinting attack (Pham et al., 2019). To perform a QoS attack, a malicious app on the victim's device creates a sticky background service. The malicious app only requires permission to use stats from the user. A cache profiling tool is then run to obtain spanning addresses to perform the exotic atomic operations. Upon detecting the victim app, the Exotic Atomic operations start and degrade the QoS of the victim's app. This loop keeps on running until the app is not in the foreground. Then, the QoS degradation attack is stopped, and the system bottleneck is released as soon as the user quits the victim app. This procedure keeps on repeating until the app is removed from the phone. Sensitive app fingerprinting attack (Pham et al., 2019) also works in a similar fashion to QoS attacks. In a sensitive app fingerprinting attack, There are multiple ways in which a malicious app, depending on its permissions and privileges, can uncover what other app installed on a victim's device. For example, a malicious app can easily check if a specific target app is installed on the victim's device by using specific API calls, access to the device storage or a VPN service. With some special permissions, a malicious app can get the list of running processes or infer the UI states. With debugging privilege, the app can retrieve the list of package names and learn the path to the installation file of a specific target app. Further, a curious app can also achieve this by using multiple API calls.

8.7. Software & Hardware Tools, Mobile Apps, Permissions & Human Capabilities

Interfaces, e.g., for financial transactions, can be maliciously targeted when software tools are used in addition to a mobile app with access to permissions and hardware tools and human capabilities such as UPI-based attacks (Kumar, Kishore, Lu, & Prakash, 2020). Multiple attacks could be performed on payment interfaces, such as unauthorized registration by using a user's phone number, unauthorized bank transactions using the victim's phone number, and partial debit card number, and unauthorized transactions without the debit card number. To perform these attacks, the attacker uses a rooted phone and reverse engineers the payment apps. Debug statements are then added and repackaged with signature statements from the attacker. The attacker then releases the repackaged version as a malicious app that requests Internet access and access to SMSes and the phone state. The victim downloads the app and grants the permissions. Then, by following the standard procedure of signing up and granting permissions, the attacker reveals sensitive information and is able to perform malicious activities.

9. Level 4: Expert Attacks

The attacker in this category is a resource-rich adversary that makes use of various combinations of resources, such as mobile phone apps, software tools, advanced programming, hardware tools, and access to phone permissions. In some cases, manual tools and human capabilities are also required. The requirements make the attacks sophisticated and require expert knowledge to implement them. Hence, such attacks can be labelled as "Expert Attacks". Considering ISO-IEC standards, this attack corresponds to **SL4**.

9.1. Advanced Programming

Only through advanced programming expertise, an attacker is well-equipped to perform an attribute inference attack (Jia & Gong, 2018). To perform an attribute inference attack, the attacker uses public data (such as review data) and a machine learning classifier to get a victim user's private attributes, such as location. The machine learning classifier is a multi-class classifier that takes review data as input, and by using a training dataset gets the city lives of the victim. The public data is easy to collect and can be found on public profiles, such as social network profiles.

9.2. Advanced Programming & Mobile Apps

Equipping a mobile application with advanced programming can result in inference attacks and microarchitectural attacks (Frigo, Giuffrida, Bos, & Razavi, 2018). In an inference attack (Spreitzer, Kirchengast, Gruss, & Mangard, 2018; Spreitzer, Palfinger, & Mangard, 2018), there are two phases; training and attack. In a training phase, the attacker builds templates for information leaks, e.g., by collecting API calls and dynamic time warping. In the attack phase, the attacker distributes a malicious app that does not require any permission. The app observes identified information leaks and infers corresponding events. This type of attack is similar to the activity interface inference attack (Yang et al., 2019) detailed above, yet requires different resources and reveals other kinds of information. Likewise, microarchitectural attacks (Frigo et al., 2018) steal data using a diverse range of side channels or corrupt data using hardware vulnerabilities. The attacker has access to an integrated GPU either by deploying a malicious app or directly through malicious scripts when a user visits a website. The attacker only makes use of primitives of the GPU.

9.3. Advanced Programming, Hardware Tools & Human Capabilities

Reflections of a virtual keyboard can be compromised to leak sensitive information with the aid of advanced programming, hardware tools, and human capabilities. Such attacks are referred to as reflection-based attacks (Raguram, White, Goswami, Monrose, & Frahm, 2011). In this attack, not just capturing a video of the virtual keyboard of the victim's phone but also the reflection of the virtual keyboard on reflective screens, such as the victim's sunglasses, could reveal what the victim typed. This attack requires the attacker to be somewhere near the victim and video-record the user interaction through a video recording device either by directly observing the screen or by observing the reflections of the screen in nearby objects. The attacker can also install a video recording device in the victim's environment to minimize the attacker's noticeability. After the video recording, the attacker works on acquiring stable frames of the video sequence which are then used for stabilizing image transformations. Next, the video frames are aligned against a reference image of the victim's phone. The attacker then trains the classifiers to detect the keypresses made by the victim. The output is refined by building a language model that also serves the purpose of filling in the missed detections.

9.4. Advanced Programming, Mobile Apps & Hardware Tools

The timing of sensitive user interfaces can be leaked to perform a power analysis attack. To perform such an attack, the attacker requires advanced programming, a mobile app, and hardware tools. An example of such an attack is the power analysis attack (Guo, Ma, Wu, & Chen, 2018). A power analysis attack requires a malicious app to be installed in the OS environment in which the victim's app is running. The goal of the attack is to know the timing of the sensitive UIs as they appear on the smartphone screen. The attacker then performs the next steps to disclose the confidential information. When the malicious app is in the process of misusing the power side channel, it functions in the background and records the power data while the target app is running in the foreground. The malicious app tries to infer the sensitive UI of the victim app based on the collected power traces. After the identification of the target UI, further attacks can be carried out.

9.5. Advanced Programming & Software Tools

A combination of advanced programming and software tools can be used to perform severe attacks such as a memory disclosure attack (Gruss, Bidner, & Mangard, 2015). Unlike most attacks, this attack does not require the installation of a malicious app on the victim's device. All it requires is for the victim user to visit a website that contains the attacker's malicious code. Through this, the attacker can uncover which apps run on the victim's device, user activities, and the specific web pages open on the victim's device. This attack has one condition to be successful, i.e. page deduplication should be enabled on the smartphone. It then exploits page deduplication to perform a memory disclosure attack. This attack has 3 steps: 1) filling a page with expected data to be found on the victim system through malloc implementation, 2) waiting for the operating system or hypervisor to deduplicate the attack arrays, 3) and lastly measuring the write-access time to know whether a page has been deduplicated.

9.6. Advanced Programming, Software Tools & Hardware Tools

An amalgamation of software and hardware tools with advanced programming can be leveraged by an attacker to perform attacks like a remote screen attack (Z. Li et al., 2020) or mobile social network attack (Ometov et al., 2017). Through a remote screen attack, it is possible to exploit the screen display. The victim merely needs to have a website containing a malicious script. This attack works by exploiting the display mechanism using liquid crystal (LC) elements that act as a passive signal modulator and LCS response that contains screen information. An RF signal processing scheme, including a deep learning model, assists in the wavelet analysis, which is then followed by the spectrogram feature augmentation. Based on this concept, Li et al. (Z. Li et al., 2020) modelled a proof-of-concept by developing "WaveSpy" - a remote screen inference system that uses mmWave-based LCS response to get real-time sensitive information without any knowledge of the screen and that too through the wall.

Along the same lines, a mobile social network attack aims to get user traces from a smartphone with the use of an external sound card. To perform this attack, devices such as Alcatel POP3 are needed. In the training phase, a clean run utilizing sandbox proposed by (Ometov et al., 2017) is executed. Then, the traces are analysed. The next step involves raw data pre-processing and is done for the subsequent neural network analysis. Following this, trace synchronization is achieved. The attackers are then able to detect crypto computation signals.

9.7. Advanced Programming, Software Tools, Hardware Tools & Human Capabilities

When advanced programming, software and hardware tools are merged together with human capabilities, attacks like advanced smudge attacks (Cha, Kwag, Kim, & Huh, 2017; Shin, Sim, Kwon, Hwang, & Lee, 2022), and website fingerprinting (Spreitzer, Griesmayr, Korak, & Mangard, 2016) or video-based attacks (T. Chen, Farcasin, & Chan-Tin, 2018; Shukla & Phoha, 2019; Ye et al., 2017; Yue et al., 2014) can be easily done to invade users' privacy and bypass security.

Advanced smudge attack requires examination of smudges left on the device screen after it has been used by the victim. This requires physical access to the victim's device. To analyze the smudges, attackers can use (1) image processing to infer possible patterns from the smudges by pre-processing (Canny, 1986; Matas, Galambos, & Kittler, 2000), e.g., by OpenCV (OpenCV, 2022b), and (2) sorting patterns based on the occurrence probabilities computed using n-gram Markov model which is built using real-world pattern data-sets or using deep learning. Similarly, website fingerprinting attacks could be performed with the help of a malicious app running in unprivileged mode and monitoring incoming and outgoing traffic statistics from tcp_rcv and tcp_snd of a target app. This data acts as training data. After collecting the data, the malicious app looks for relaunching of the target app. It then gathers traffic data from tcp_rcv and tcp_snd again and matches the collected data with previous training data to infer the sensitive information. The collected data could also be sent to a remote server for analysis and for matching the device name of the attacked device with the training devices available

Another way to steal authentication credentials using the same set of resources is to observe and video capture the hand movements of the victim when they are typing the password (Shukla & Phoha, 2019) or unlock pattern (Ye et al., 2017). The video could be captured using any video recording device, such as a smartphone, a camcorder, or through surveillance camera footage. The resulting footage can be analyzed by different means, including TLD Tracking Tool (Kalal, Mikolajczyk, & Matas, 2011) for tracking the anchor hand point and the anchor point on the apparent side of the mobile device, edge detection (Von Gioi, Jakubowicz, Morel, & Randall, 2008), and further computer vision tools, to reconstruct the user input. For example, the attacker can build a probability-based password model using two large data sets: 1) UNIQPASS v15 Password Data set (N. E. Security, 2022), and 2) Video data set for computer vision analysis.

9.8. Advanced Programming, Software Tools & Mobile App

Treacherous attacks such as cross-cache (Lipp, Gruss, Spreitzer, Maurice, & Mangard, 2016) or flush-reload side-channel attacks (X. Zhang, Xiao, & Zhang, 2016) can be performed by combining advanced programming with software tools and a mobile app. Cross-cache attacks require a malicious app that does not prompt any permission. This attack can monitor the activity of the GPS sensor, camera or Bluetooth. This information leak can help the attacker to know details about the victim. In a learning phase, a template matrix is computed to see how many cache hits occur on a specific address. Then, in the attack phase, this matrix is used to infer events from the cache hits. The events could be stimulated via the android-debug bridge (adb shell).

Using the same resources as required by cross-cache, flush-reload side-channel attacks make use of a malicious app packaged together with a native component that is compiled with Android NDK. The attacker is equipped with the knowledge of C and C++ programming languages.

9.9. Advanced Programming, Software Tools & Permissions

The inviolability of user's privacy can be bypassed by exploiting AdSDKs attack (Son, Kim, & Shmatikov, 2016), which is a fusion of advanced programming, software tools and phone permissions. Not just malicious apps but also malicious ads displayed can infer sensitive information about users by accessing external storage. The most important asset in this attack is an ad-supported app that runs on the target user's device and shows malicious ads in a confined WebView instance. For instance, the attacker can trick the victim into downloading an HTML page that holds a malicious payload. After the payload page is presented to the user, the attacker's ad calls the payload by opening this page within the same WebView where the ad is running.

9.10. Advanced Programming, Software Tools, Hardware Tools & Mobile Apps

Combining advanced programming with software and hardware tools and mobile apps can enable complex attacks, such as power analysis or inference attacks on processors.

An inference attack (Gulmezoglu et al., 2019) on the processor requires a permissionless malicious app to be installed on the victim's device. The attacker can acquire knowledge of running apps, launching websites, and streaming videos. In the training phase, the attacker builds a machine learning/deep learning model on a training device similar to the victim device by recording the raw LLC profiles of target apps, websites, and videos. The trained models are then integrated into the app and published on the app store. In the second phase, the malicious app prepares eviction sets for profiling the LLC on the target device, followed by extracting vector features. They are then classified with already trained models to infer sensitive content, including opened applications, websites, and streaming videos.

Parallel to inference attack on the processor, power analysis side-channel attack (Yan, Guo, Chen, & Mei, 2015) uses the unprivileged power consumption traces, to infer sensitive UIs, guess password lengths, and also estimate geolocations. A malicious app running in the background collects power traces continuously. The power patterns can be collected either through hardware-based methods (e.g., a Monson Power Monitor) attached to the target smartphone or through software-based methods (e.g., directly polling voltage and current readings within the mobile system). The collected power traces can then be analysed to infer confidential data. A malicious app's key role in this scenario of exploiting PSCs is to achieve automatic detection of pre-learned power patterns. This can be achieved by pattern matching or machine learning algorithms, e.g., dynamic time warping (DTW).

9.11. Advanced Programming, Software Tools, Hardware Tools, & Permissions

Signal reflection information can be targeted with the help of advanced programming, software and hardware tools, and permissions. A wireless transmitter-based attack (J. Zhang et al., 2016) collects signal reflection information before the user starts to unlock a device until the user ends up unlocking the device. In (J. Zhang et al., 2016), researchers proposed an approach to performing this attack called; WiPass. They collected CSI data and used discrete wavelet decomposition to remove noise from obtained signals. (1) WiPass removes the noise from collected signals using a two-level Symlet filter, (2) uses the DCASW to extract the features to build the finger motion profiles and finally as the last step, (3) uses a hierarchical dynamic time warping (DTW) approach to recognize the unlock passwords.

9.12. Advanced Programming, Software Tools, Mobile Apps & Permissions

Attacks similar to GUI squatting (S. Chen et al., 2019), gesture typing (Simon, Xu, & Anderson, 2016), and keystroke attacks (Owusu, Han, Das, Perrig, & Zhang, 2012) are feasible to perform if a mobile app and permissions are complemented with advanced programming and software tools.

In a gesture typing inference attack (Simon et al., 2016), gesture typing keyboards are the target. It involves a malicious app running with Internet access. The malicious app observes and records publicly available events from the system while the user enters text in the victim app. The malicious app can only record the signals, i.e. the counters, but not the words themselves. For each word entered by the user, a series of events is observed in the system that can be used as a fingerprint to recognize the word entered. Supervised machine learning (Recurrent Neural Network (RNN)) is used to remove noise from the data. The fingerprint is constructed from the training data and is used to infer sentences entered later in the victim app. The RNN outputs for each word signal in a sentence signal a probability that the word signal corresponds to a particular word in the dictionary. The data is sent to remote attackers.

GUI squatting attack (S. Chen et al., 2019) refers to automatically generating phishing apps using image processing and deep learning techniques. The automatically generated phishing apps have the ability to steal sensitive information by taking screenshots of login screens. The generated apps require Internet permission to upload the collected sensitive information to a remote server. To generate phishing apps, image processing techniques, such as canny edge detection (OpenCV, 2022c) and edge dilation (OpenCV, 2022a) are used. The GUI components are classified with a deep learning algorithm, i.e. a CNN. Then, these components are arranged to generate layout code matching the XML file for the imitation of the original apps. Then, the deception code is designed for the interactive components, and a response is assigned to each interactive component. Chen et al (S. Chen et al., 2019) implemented this approach using Python and several open-source libraries, such as OPENCV (OpenCV, 2022b) and OCR techniques (OCR, 2022).

Keystroke attacks through accelerometer readings (Owusu et al., 2012) involve a malicious app running in the background collecting accelerometer readings. By using machine learning (e.g., Random Forest Algorithm (Ho, 1995)), the text entered on a device is extracted from accelerometer readings. The app requires network access for uploading the collected data and access to fine-grained accelerometer data. This attack involves keypress segmentation, probabilistic keypress classification and sorting keystroke sequences by maximum likelihood. A probabilistic error model is constructed for sorting keystroke sequences by maximum likelihood.

In a Pin Skimmer attack (Simon & Anderson, 2013), the user installs a malicious app and has root access to the device. This attack requires access to the camera and microphone. A smartphone with two operating systems (e.g., Android and TrustZone OS) that operate in parallel are required to perform this attack. The malicious app cannot access sensitive information available on the TrustZone OS, even with root access on Android OS. The sensitive information apps are launched in the Trusted OS. The rootkit retains access to certain shared resources like an accelerometer, camera, GPS, microphone, and like. Using the front video camera and the microphone, the Pin Skimmer attack collects all user-pressed events entered into the sensitive app and records them using the front camera in a video file with audio. It saves the image to disk, and the attacker then uploads the collected data to a remote server where, through image processing skills, the exact PIN is retrieved. Support Vector Machine (SVM) (Zisserman, 2015) was implemented as a learning algorithm with open source libraries LibSVM (C.-C. Chang & Lin, 2021) and Weka (at Waikato University, 2021).

A user's metro location can be inferred through a malicious app that reads accelerometer and orientation sensor data and uploads the readings to a remote server (Hua, Shen, & Zhong, 2016). This location inference attack aims to reveal what a target user's metro ride trace is. It accomplishes its mission by noting the differences between distinct station interims leading to distinct macro motion characteristics that are captured by the motion sensors of the victim's smartphone. The readings are then analysed, and machine learning algorithms are used to identify the victim's ride intervals.

Recording tap sounds and vibrations while an application is running from the stereo-microphones and gyroscopes of a smartphone, can be maliciously used to perform a keystroke inference attack (Narain, Sanatinia, & Noubir, 2014). To perform this attack, a malicious application like a custom keyboard is presented to the user to collect typing behaviour for the purpose of training a model. The microphone requires permission during installation, but this permission can be justified with the aid of any feature offered by the malicious application. After monitoring the victim's behaviour, the malicious application uploads the collected data to a remote server. After training, it listens to the keypresses in the background from the sensitive Android applications. Then the application detects the point of interest location of the victim using GPS or cellular or wireless networks, then the malicious application collects gyroscope and microphone data. A fast Fourier transform filter could be used to detect frequencies corresponding to the sample tap values.

Ambient light sensors could also be exploited to perform attacks on user input such as PINs (Spreitzer, 2014). They can be accessed via the Android Sensor API. To perform such an attack, a malicious application is used to collect the light-sensor information while the user is interacting with the device. The malicious application tricks the user into the application in a manner similar to inputting PINs. It then uses this data as the training data. Malicious applications also require an internet connection to have powerful servers for machine learning algorithms. After collecting sufficient samples, the malicious application again tricks the victim user into restarting the device or the victim application. This is done to capture the ambient-light sensor information during PIN input of the victim application. Then by means of machine learning, the PIN input is retrieved. Matlab statistics toolbox can be used to determine the PIN entered.

App fingerprint attacks (Matyunin et al., 2019) could be performed by exploiting magnetic sensor measurements to infer current activities on the smartphone. Therefore, fingerprint browsing and app activity are possible. For this, a malicious app with Internet access and access to zero-permission sensor information is required. To fingerprint the browser, the victim opens a webpage that is controlled by the attacker. The webpage has at least some malicious component belonging to the attacker, such as ads. Magnetometer readings are collected continuously and the attacker attempts to identify the launched apps or websites with the help of a supervised learning approach. The malicious app gathers the labelled traces for all websites and apps. The learning could be performed on numerous devices that the attacker holds or accessed using cloud testing platforms. In the case of website fingerprinting, the learning phase could also be done on the victim's phone. Principal component analysis is performed on the magnetometer data, and random forest is used to classify the traces.

An inference attack through interrupt timing attack aims to discover the unlock pattern or sensitive information entered by the target user (Diao, Liu, Li, & Zhang, 2016). Diao et al. (Diao et al., 2016) proposed a novel way of doing this by tricking the target user into installing a malicious app. The malicious app requires no permission from the user as it works by reading interrupt statistics which are public to any process and contain information about all running devices. This information is used to infer sensitive information passing through the running devices. This attack can collect two types of sensitive information, unlock pattern and UI information. After collecting the unlock pattern information, it could be uploaded to a remote server for which INTERNET permission will be required. The UI information could be used for further malicious attacks such as phishing. In Diao et al.'s implementation of the attack, they used native C and Java with Android NDK (Android, 2022) to write the interrupt modules for the malicious app. They then trained a Gaussian model using data from 5 participants to infer the sequence.

Textual content can be easily leaked through a malicious application (Amiri Sani, 2017). The malicious application can compromise the OS and achieve root or kernel privileges. A malicious application such as malware can use various methods to achieve root privilege, such as a rowhammer attack. For kernel privileges, malicious applications can make use of code injection or return-oriented programming. The malicious applications can then use ADB capability to store screenshots. Wei et al. (Amiri Sani, 2017) presented SchrodinText as a solution to protect specific textual content decided by the application developer.

User input inference attack (Ulqinaku, Malisa, Stefa, Mei, & Čapkun, 2017) requires a malicious app that runs in the background and records all hover inputs of all apps. The malicious app has access to SYSTEM_ALERT_WINDOW and the Internet. The data collected is uploaded to a remote server for analysing the data.

A UI state inference attack requires an app running in the background and access to the Internet (Q. A. Chen, Qian, & Mao, 2014). The attack first detects the activity transition event, which is known by the shared-memory side channel. After the detection, the identity of the new activity is achieved using the Activity signature and Activity transition graph. In a training phase, an automated tool is built to generate

Activity transitions in an app and collect feature data to build the activity signature and the activity transition graph. In the attack phase, the app collects feature data during activity transitions. It then leverages the activity signature and a transition model based on the activity transition graph to execute the attack.

Train routes can be identified by exploiting device sensors, such as accelerometer, magnetometer, and gyroscope. Such attacks can be classified as sensor-based positioning attacks (Watanabe, Akiyama, & Mori, 2015). As a first step, a machine learning algorithm is applied to the sensor data, and then the activity of the user is detected. Next, the departure and arrival times of vehicles from the sequence of human activities are detected. Finally, by correlating the detected departure and arrival time of the train with the aid of timetables and route maps, the potential route of the journey is identified. This whole process primarily requires a malicious app with internet access on the victim's device. The malicious app continuously collects sensor data and sends them to the adversary who then estimates the route of travel by analysing the sequences. The attacker holds information on the list of public transport systems that are likely to be used by the victim. Machine learning, specifically random forest, is used to process the sensor information.

A digital password inference attack (Tang, Wang, Wang, Zhao, & Wang, 2018) leverages an accelerometer to reveal passwords on smartphones by exploiting the user-independent features of the movement of tapping buttons. Angle features are extracted to reflect changing trends and a multicategory classifier by combining the dynamic time-warping algorithm to get the probability of each movement. Then, by using a Markov model, the unlock process is modelled, and the sequences with the highest probability are used as the attack candidates. The data is sent to a server, which cleans it from noise and segment movements. Then, the data is used to train a classifier. It is then combined with the dynamic time-warping (DTW) algorithm to reveal the possibility and probability of each movement of a password sequence. The Markov model is then used in the unlocking process with multiple movements.

User-typed text can be extracted by recording the sound of the in-built microphones of a smartphone through an acoustic emanation attack (H. Gupta, Sural, Atluri, & Vaidya, 2016). Signal processing techniques assist in extracting a probable set of characters per tap, and then by using natural language processing algorithms, most probable words and sentences are constructed. From the recorded audio signals, the first step is to detect tap instants which can be done by using the Detect Peak Intervals algorithm. It takes audio as input and returns a set of time intervals as the output. A malicious application installed on the victim's device can easily record the audio and later send it to a remote server for processing. The malicious application only needs permission to access the internet and a microphone.

9.13. *Advanced Programming, Software & Hardware Tools, Mobile App, & Human Capabilities*

An attacker can learn about the victim's path using advanced programming with software tools, a mobile application, hardware tools, and human capabilities. In a user path identification attack (Lakshmanan, Budhdev, Kang, Chan, & Han, 2021), the attacker identifies the walking path of a user by connecting the real-world identity to the network identity of the devices. To practically implement this attack, the attacker uses a low-cost software-defined radio device, such as USRP (Ettus, 2022) with open-source cellular projects, such as srs LTE (srsRAN, 2022). This attack requires two mandatory steps: (1) the attacker has to be located within 0.4-2 kilometres of the victim, and (2) the user must carry out a mobile downlink activity while walking, e.g., streaming a video. The adversary must have some basic knowledge about when the victim is walking, and that is when the attacker performs the path identification. As the victim accesses mobile downlink activity, the attacker captures the number of secondary cells at each location.

9.14. Advanced Programming, Software Tools, Mobile App, Hardware Tools & Permissions

A fusion of advanced programming, software, hardware tools, and phone permissions can lead to keystroke inference (Ping, Sun, & Mao, 2015), a smartphone speech inference (Griswold-Steiner, LeFevre, & Serwadda, 2021), and sensor-based location tracking attacks (Narain, Vo-Huu, Block, & Noubir, 2016).

A keystroke inference attack (Ping et al., 2015) considers short inputs by the users, such as PINs or passwords and long inputs, such as emails or text messages. To perform this attack, a malicious app must be installed on the target user's device. Data from acceleration and gyroscopes are collected and used as training data. All collected data is temporarily stored on the SD card of the mobile device and is transferred to a remote server as soon as the phone is connected to the WiFi. The app only requires access to four user permissions: INTERNET, READ_PHONE_STATE permission, WRITE_EXTERNAL_STORAGE, and GET_TASKS permission. This attack assumes that the target user is using the standard QWERTY soft keyboard in a vertical orientation on his device.

The smartphone speech inference attack (Griswold-Steiner et al., 2021) is performed during a phone conversation by a malicious app having access to the motion sensor with the aim of making inferences on the voice content of the phone conversation. To conduct classifier training for speech inference attacks, the attacker can use a target-agnostic (TAG) or target-aware (TAW) approach. In TAG, the attacker collects training data from the accomplice who speaks words of interest on the phone while collecting the accelerometer and gyroscope data. In TAW, the training set would include data collected from the target. To discover the hardware used by the victim, the attacker may use surveillance in person or use recorded video that reveals how the victim holds the phone. This information trains the deep neural network for the speech inference attack.

Sensor-based location tracking attack (Narain et al., 2016) is performed when a victim user is driving a car with a smartphone. This attack uses smartphone sensors and tracks the victim user's location. To perform this attack, an app must be installed on the user's device that collects sensor data: accelerometer, gyroscope, and magnetometer. The recorded sensor data is uploaded to a remote server and processed. Turn angles, route curvatures, accelerations, headings and timestamps data are combined with public geographic area information to infer the user's route. This process is facilitated by graph construction and a search algorithm.

9.15. Advanced Programming, Software Tools, Mobile Application, Hardware Tools, Permissions, Human Capabilities

Not just outdoor location but the indoor location could also be inferred when advanced programming and software tools are added to a mobile application along with permissions, hardware tools, and human capabilities. An indoor location inference attack (Zheng & Hu, 2019) includes a malicious app that secretly collects sensory data, including accelerometer, gyroscope, and magnetic field sensors and, in return, eavesdrops on the location. The app requires access to a network, either WiFi or cellular, to upload the location information to the attacker's remote server. In a training stage, the attacker walks through the targeted indoor location while carrying a number of mobile devices. This way, they collect the sensor readings as they pass through the targeted location track. To improve the accuracy, Bluetooth Low Energy (BLE) beacons are used in each sensitive location to activate sensor readings automatically as the attacker passes through it. Then, segmentation is performed on the large length of the data stream to get the desired specific part of the data stream, known as the exemplar. Further, noise reduction is performed. After the exemplars are ready, a robust supervised learning scheme using an anomaly calibration technique is used to construct a classifier to recognize the sensor pattern for each sensitive location. In the attack phase, the attacker adds the classifier to a malicious app, which then collects the sensor readings in the background and sensitive indoor locations.

10. Discussion

In this paper, we propose categorising social engineering and side channel attacks based on the resource-oriented nature of attacks. While several vectors could indicate the feasibility of an attack, such as the cost of resources, this is easily overcome due to the availability of multiple vendors where an attacker can get access at a low or cheaper price. For example, a thermal camera can be brought from Amazon at a price like £155 (Amazon, 2023) but can also be bought cheaper from places like eBay or Facebook Marketplace. This makes the cost of resources slightly less attention grabber, and access to resources is the top priority for the investigation to determine the feasibility of attacks.

10.1. Using the Categorization

The proposed categorization presented in this paper can be used in multiple ways. Below, we discuss a few usage directions and research questions that the categorization can assist in answering.

1) Attack Assessment: Our categorisation can be used as an assessment method to ease the carrying out of specific attacks. In doing so, we can estimate the share of the population capable of executing a particular attack, which would indicate the ubiquity of the attack. For example, carrying out a novice attack, such as a traditional shoulder surfing attack, would only require the attacker to be in close proximity to the victim and make close observations. In contrast, performing an expert attack, such as a GUI squatting attack, requires more sophisticated tools and skills, such as image processing and deep learning. Comparing the resources required for these two attacks shows that anyone, regardless of background and expertise, could be a shoulder surfer as seen in prior work as well (Eiband et al., 2017; Farzand, Marky, & Khamis, 2022), but to make an expert attack such as a GUI squatting attack, one has to be well-equipped with tech and security knowledge and tools. In sum, the low barriers to invading someone's privacy make it possible for a more significant proportion of the population to become attackers with little to no training. This also points out that similar attacks could occur anywhere at any time, heightening the need for adequate mitigation. Our categorisation would help organizations and individuals to set defence priorities and make informed decisions when using smartphones in different environments, such as private or public. Furthermore, the proposed categorisation can also assist organizations in making informed decisions about resource allocation when developing policies and methods to mitigate specific attacks. It can also be used to classify the severity of new emerging attacks. First, a list of requirements is required to carry out the attack. Based on the requirements, the attack can be linked to one of the four categorisation layers (see Section 4). We now present an exemplary thought to showcase how the categorisation can assist individuals, researchers, and organizations in conducting attack assessments using the proposed categorisation.

Example 1: Utilising Individual of Categorisation Novice attacks are at the centre of attention as they require minimal resources that anyone can easily acquire. Using categorization to understand the attack requirements sets the focus on limiting access to resources or prohibiting their use in unavoidable circumstances. For example, shoulder surfing is a type of novice attack that only requires one to be in close proximity to the user and make careful observations. This property of shoulder surfing attacks makes them practical at any place around anyone. From the perspective of an individual, the individual knows that the attack is performed through direct observation of the screen, so the individual will be careful when accessing personal information in the vicinity of others in public and private environments. This behaviour could vary among users as they vary in their perception of the importance of privacy (Farzand et al., 2024). From the perspective of researchers, they can investigate the core requirement of the practicality of the attack, i.e., screen observation. For example, the details on the angle or duration of observation and distance between attacker and victim could assist in designing adequate countermeasures (Abdrabou et al., 2022; SAAD, LIEBERS, GRUENEFELD, ALT, & SCHNEEGASS, 2021). Furthermore, organizations could propose policies that prohibit access to sensitive information in public environments or set conditions for access.

Example 2: Utilising Layers of the Categorisation Focusing on specific layers of the categorisation could help researchers in designing holistic protection methods. By inspecting the common requirements in each layer of the categorization, common resources can be extracted, and then countermeasures specifically targeting the availability and use of those resources could be limited or prohibited as per the scenario. This would help in providing holistic protection against a group of attacks.

2) Accessibility versus Scalability: As we move horizontally across the categorisation levels, the feasibility of an attack decreases as the complexity of resources required increases. For example, to perform Expert Attacks, attackers must have advanced programming knowledge, software tools, mobile applications, access to user phone permissions, and hardware tools. However, the attacks at the Novice Level require human capabilities and easily available hardware tools, which can be performed more easily. This introduces interesting scalability aspects: the more difficult it is to execute an attack, which makes it less accessible, the more potential victims can be targeted.

While Novice Attacks that don't require technical skills or special equipment or setup, are easily accessible to anyone, executing the attack does not scale well because one attacker can only target a small number of victims at a given instance, mostly only one. For example, in the case of a shoulder surfing attack that requires observing someone's device screen without permission, an observer can only observe one screen at a time. Similarly, thermal attacks, which are another example of Novice Attacks, can be done on one user's device at a given instance. On the other hand, more sophisticated attacks scale better since one attacker can target many users. For example, juice filming attacks, which are examples of proficient attacks, require a one-time setup, and then multiple users could be the target. One might argue that this is an advantage because there are hurdles to overcome in order to become a large-scale attacker. Yet, research also hints at another issue: easily available attacks might not be well-known by potential victims, and hence, they may be more susceptible to suffering the attack's consequences. For example, in a study by Jiang et al., (Jiang et al., 2017) 74.5% of participants did not know about charging attacks, but only 14.1% of participants did *not* know about malware-based threats. Therefore, charging attacks might become more prevalent than malware-based attacks because of 1) the easy setup and 2) the lack of user awareness. This might be similar to other attacks that can be easily executed.

10.2. Key Takeaways & Future Research Directions

1) Anyone can easily become an attacker. Our categorisation has four different layers on how difficult it is for attackers to execute the attacks, which indicates the attack's scalability. Attacks in the "expert" layer require sophisticated knowledge and resources. Even though these attacks can scale well, they are unlikely to become ubiquitous because the hurdle for attackers is too high. "Proficient" attacks are on the verge of being script kiddies by using malware available online and programming skills. What is more concerning, though, are the lower two layers, "novice" and "intermediate." Attacks in the layer "intermediate attacks" require less expertise, some hardware tools that can be bought easily, and a mobile application that can be available online. Each requirement is benign and, hence, easy to get (e.g., video editing software). Consequently, this level can be reached by individuals with low knowledge, drastically reducing the hurdle to becoming an attacker. Finally, "novice" attacks like shoulder surfing do not require technical expertise and setup. Anyone can become a shoulder surfer spontaneously, and probably most individuals have already shoulder-surfing someone even without intention (Eiband et al., 2017; Farzand, Bhardwaj, Marky, & Khamis, 2021; Farzand, Marky, & Khamis, 2022). Consequently, carrying out "Novice" attacks is no longer restricted to highly motivated criminals with specific resources, *anyone can now become an attacker.*

2) Individual mechanisms are insufficient. Many attacks exist to target various attributes of mobile user privacy and security; the literature also underpins numerous mitigation or protection methods. For example, for protection against shoulder surfing attacks, a user can utilize mitigation methods, such as EyeSpot (Khamis, Eiband, Zürn, & Hussmann, 2018). Similarly, for protection against thermal attacks (Alotaibi et al., 2023; Marky, Macdonald, Abdrabou, & Khamis, 2023), mechanisms such as PIN scrambler (Kirkwood et al., 2022) can be used. The problem with using such individual mechanisms is that they require extra effort from the users and more time, rendering them ineffective (Gugenheimer et al., 2015; Harbach, Von Zezschwitz, Fichtner, De Luca, & Smith, 2014; Khamis et al., 2016; Krombholz, Hupperich, & Holz, 2016). Individual mechanisms also need memory allocation on the devices and have specific device model requirements to fulfil for the user to use the mechanism. In such a situation, what matters the most is how non-expert users can protect themselves and minimize the possibility of being attacked without additional protection mechanisms that require much effort. This demands a more holistic understanding of user protection focused on an entire attack ecosystem rather than patching devices to resist single attacks. Further, mobile devices combine more and more functions ranging from shopping to banking that users want to perform on the go. As these devices can be attacked more and more easily, they result in a single point of failure that is not well enough protected.

Future Research Direction #1

Q1: How can users be ubiquitously defended against groups of attacks rather than patching against individual attacks?

Future Research Direction #2

Q2: How can we improve security and privacy mechanisms on mobile devices to safeguard them better?

3) User awareness alone is insufficient. Attacks can happen anywhere in the physical and digital world without time constraints. Awareness of the user's surroundings has repeatedly been proposed as a possible solution to protect users against multiple attacks without the need to have an additional mechanism in practice (Cha et al., 2017; Eiband et al., 2017; Meng et al., 2015; Raguram et al., 2011; Yu et al., 2017; J. Zhang et al., 2019). While this might be a viable solution in some situations, (e.g., using a public WiFi), we cannot expect users to be aware of all possible attacks whenever they use a mobile device. Further, monitoring surroundings requires much too much effort from the user and could result in a waste of interaction time with the device. Furthermore, much of the surrounding awareness goes unnoticed because of the cognitive load caused by the task the user is performing on the device, for example, in the case of shoulder surfing (Goucher, 2011). Because of that, we need viable alternatives to defend users who do not rely on users to pay attention and defend themselves.

Future Research Direction #3

Q3: How can we effectively defend mobile users in their daily lives without relying on their awareness of their surroundings?

4) Rethinking the app developer's role in providing protection. While the non-expert and expert users play their part in protecting the privacy and security of their mobile phones, app developers can contribute by making app-level improvements. For example, changing the grid pattern location can assist against smudge attacks (Cha et al., 2017). Adding body noise while using public WiFis can help with location-based attacks (J. Zhang et al., 2019), restricting access to certain proc files can safeguard against UI state inference attacks (Q. A. Chen et al., 2014), forcing apps to declare the purpose for accessing mobile phone sensors and adding noise to the sensor data can protect against sensor-based attacks. However, most attacks require access to the Internet only to implement an attack successfully. The INTERNET permission is marked as safe permission by Android (for Developers, 2022; H. Security, 2022) and is granted to apps without asking the user. Attackers can exploit this privilege to upload the collected sensitive information to a remote server for processing using advanced programming skills, such as machine learning. Second, most attacks target user location. While the location is extremely important information that enables users to accomplish various tasks, it is most compromised. Location data can be preserved by anonymization, but attacks on anonymization have also been witnessed (Golle & Partridge, 2009). The security incidents of location leakage might be one reason users are reluctant to adopt COVID-19 Contact Tracing Apps (Farzand, Mathis, Marky, & Khamis, 2022).

Future Research Direction #4

Q4: How can the developers be helped to configure Internet access to make it hard to exploit for performing attacks?

Future Research Direction #5

Q5: How can location privacy be better preserved?

11. Conclusion

With the increasing ease of access to resources to perform attacks, the security and privacy of mobile phone users are at risk. This paper explores the resources required for an attacker to carry out an attack. Based on the latest literature and a sample of 65 papers, we present a multi-layered categorisation of social engineering and side-channel attacks on mobile phones. The categorisation provides evidence for how user privacy can be violated with as little effort as direct observation through using human capabilities and as enormous effort as combining installing a malicious app with advanced programming skills, hardware tools, and much more. By analysing the work surveyed, we conclude with future research directions to better protect the privacy and security of mobile phone users.

Acknowledgements and Funding

This work was supported by an Excellence Bursary Award from the University of Glasgow and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972. We would also like to acknowledge the EPSRC New Investigator Award (grant number EP/V008870/1) for their support.

About the Authors

Habiba Farzand is a PhD candidate at the University of Glasgow. Her research focuses on the intersection of human privacy, security, and interaction with technology.

Her research utilizes creative methods to identify problems and propose solutions for users' privacy and security concerns.

Melvin Abraham is a PhD candidate at the University of Glasgow. His work is in human-computer interaction, usable privacy, and augmented reality. He is interested in exploring data access methods for making more informed privacy decisions in augmented reality.

Stephen Brewster is a Professor of Human-Computer Interaction (HCI) at the University of Glasgow's School of Computing Science. He leads the Multimodal Interaction Group, which conducts world-leading research in HCI. His research focuses on multimodal HCI, using multiple sensory modalities and control mechanisms to create natural interactions between humans and computers.

Mohamed Khamis is a Reader (Associate Professor) at the University of Glasgow's School of Computing Science, where he leads the SIRIUS Lab, focusing on research at the intersection of Human-Computer Interaction and Human-centered Security. He regularly publishes in CHI, TOCHI, IMWUT, CSCW, USENIX Security and PETs.

Karola Marky is a Professor at Ruhr Bochum University, where she leads the Digital Sovereignty Lab. Her research contributes to Human-Computer Interaction (HCI), focusing on the intersection of cybersecurity and privacy with human factors and into the daily routines of individuals.

References

62443, I. (2022). *Iec 62443.* Retrieved from https://en.wikipedia.org/wiki/IEC_62443

Abdrabou, Y., Rivu, S. R., Ammar, T., Liebers, J., Saad, A., Liebers, C., ... others (2022). Understanding shoulder surfer behavior and attack patterns using virtual reality. In *Proceedings of the 2022 international conference on advanced visual interfaces* (pp. 1–9).

Adams, A., & Sasse, M. A. (1999). Users are not the enemy. *Communications of the ACM*, *42*(12), 40–46.

Aldawood, H., & Skinner, G. (2019). A taxonomy for social engineering attacks via personal devices. *International Journal of Computer Applications*, *975*, 8887.

Alliance, I. G. C. (2020). *Quick start guide: An overview of isa/iec 62443 standards security of industrial automation and control systems.* Retrieved from https://gca.isa.org/hubfs/ISAGCA%20Quick%20Start%20Guide%20FINAL.pdf

Alotaibi, N., Williamson, J., & Khamis, M. (2023). Thermosecure: Investigating the effectiveness of ai-driven thermal attacks on commonly used computer keyboards. *ACM Transactions on Privacy and Security*, *26*(2), 1–24.

Amazon. (2022). *Amazon alexa.* Retrieved from https://developer.amazon.com/en-US/alexa (Retrieved February 20, 2022)

Amazon. (2023). *Amazon Listing - PerfectPrime IR203, (IR) Infrared Thermal Imager Camera.* https://amzn.to/3H22Nod. (Accessed: 2023-01-13)

Amiri Sani, A. (2017). Schrodintext: Strong protection of sensitive textual content of mobile applications. In *Proceedings of the 15th annual international conference on mobile systems, applications, and services* (pp. 197–210).

Andriotis, P., Tryfonas, T., Oikonomou, G., & Yildiz, C. (2013). A pilot study on the security of pattern screen-lock methods and soft side channel attacks. In *Proceedings of the sixth* *acm conference on security and privacy in wireless and mobile networks* (pp. 1–6).

Android. (2022). *Android ndk.* Retrieved from https://developer.android.com/ndk (Retrieved February 20, 2022)

Apple. (2022). *Apple siri.* Retrieved from https://www.apple.com/siri/ (Retrieved February 20, 2022)

at Waikato University, M. L. (2021). *Weka 3 - data mining with open source machine learning software in java.* Retrieved from https://www.cs.waikato.ac.nz/ml/weka/ (Retrieved February 20, 2022)

Bianchi, A., Corbetta, J., Invernizzi, L., Fratantonio, Y., Kruegel, C., & Vigna, G. (2015). What the app is that? deception and countermeasures in the android user interface. In *2015 ieee symposium on security and privacy* (pp. 931–948).

Cai, Y., Tang, Y., Li, H., Yu, L., Zhou, H., Luo, X., . . . Su, P. (2020). Resource race attacks on android. In *2020 ieee 27th international conference on software analysis, evolution and reengineering (saner)* (pp. 47–58).

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*(6), 679–698.

Cha, S., Kwag, S., Kim, H., & Huh, J. H. (2017). Boosting the guessing attack performance on android lock patterns with smudge attacks. In *Proceedings of the 2017 acm on asia conference on computer and communications security* (pp. 313–326).

Chang, B., Cheng, Y., Chen, B., Zhang, F., Zhu, W.-T., Li, Y., & Wang, Z. (2018). User-friendly deniable storage for mobile devices. *computers & security*, *72*, 163–174.

Chang, C.-C., & Lin, C.-J. (2021). *Libsvm.* Retrieved from https://www.csie.ntu.edu.tw/~cjlin/libsvm/ (Retrieved February 20, 2022)

Chen, Q. A., Qian, Z., & Mao, Z. M. (2014). Peeking into your app without actually seeing it:{UI} state inference and novel android attacks. In *23rd usenix security symposium (usenix security 14)* (pp. 1037–1052).

Chen, S., Fan, L., Chen, C., Xue, M., Liu, Y., & Xu, L. (2019). Gui-squatting attack: Automated generation of android phishing apps. *IEEE Transactions on Dependable and Secure Computing*, *18*(6), 2551–2568.

Chen, T., Farcasin, M., & Chan-Tin, E. (2018). Smartphone passcode prediction. *IET Information Security*, *12*(5), 431–437.

Cho, G., Huh, J. H., Cho, J., Oh, S., Song, Y., & Kim, H. (2017). Syspal: System-guided pattern locks for android. In *2017 ieee symposium on security and privacy (sp)* (pp. 338–356).

Clarke, N. L., & Furnell, S. M. (2005). Authentication of users on mobile telephones–a survey of attitudes and practices. *Computers & Security*, *24*(7), 519–527.

Committee, I., & (TC65WG10), I. T. C. . W. G. . (2016). The 62443 series of standards: Industrial automation and control systems security.

Devi, M., & Majumder, A. (2021a). Side-channel attack in internet of things: a survey. In *Applications of internet of things* (pp. 213–222). Springer.

Devi, M., & Majumder, A. (2021b). Side-channel attack in internet of things: A survey. In J. K. Mandal, S. Mukhopadhyay, & A. Roy (Eds.), *Applications of internet of things* (pp. 213–222). Singapore: Springer Singapore.

Diao, W., Liu, X., Li, Z., & Zhang, K. (2016). No pardon for the interruption: New inference attacks on android through interrupt timing analysis. In *2016 ieee symposium on security and privacy (sp)* (pp. 414–432).

Distler, V., Fassl, M., Habib, H., Krombholz, K., Lenzini, G., Lallemand, C., . . . Koenig, V. (2021). A systematic literature review of empirical methods and risk representation in usable privacy and security research. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *28*(6), 1–50.

Do, Q., Martini, B., & Choo, K.-K. R. (2019). The role of the adversary model in applied security research. *Computers & Security*, *81*, 156–181.

Dunphy, P., Heiner, A. P., & Asokan, N. (2010). A closer look at recognition-based graphical passwords on mobile devices. In *Proceedings of the sixth symposium on usable privacy and* *security* (pp. 1–12).

Eiband, M., Khamis, M., Von Zezschwitz, E., Hussmann, H., & Alt, F. (2017). Understanding shoulder surfing in the wild: Stories from users and observers. In *Proceedings of the 2017 chi conference on human factors in computing systems* (pp. 4254–4265).

Ettus. (2022). *Usrp b210.* Retrieved from https://www.ettus.com/all-products/ub210-kit/ (Retrieved February 20, 2022)

Farzand, H., Bhardwaj, K., Marky, K., & Khamis, M. (2021). The interplay between personal relationships & shoulder surfing mitigation. In *Mensch und computer 2021* (pp. 338–343).

Farzand, H., Marky, K., & Khamis, M. (2022). Shoulder surfing through the social lens: A longitudinal investigation & insights from an exploratory diary study. In *Proceedings of the 2022 european symposium on usable security* (pp. 85–97).

Farzand, H., Marky, K., & Khamis, M. (2024). Out-of-device privacy unveiled: Designing and validating the out-of-device privacy scale (odps). In *Proceedings of the 2024 chi conference on human factors in computing systems.* New York, NY, USA: Association for Computing Machinery.

Farzand, H., Mathis, F., Marky, K., & Khamis, M. (2022). Trust & privacy expectations during perilous times of contact tracing. In *proceedings of the symposium on usable security and privacy.*

Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., & Wagner, D. (2012). Android permissions: User attention, comprehension, and behavior. In *Proceedings of the eighth symposium on usable privacy and security* (pp. 1–14).

Felt, A. P., & Wagner, D. (2011). *Phishing on mobile devices.* Citeseer.

Fernandes, E., Chen, Q. A., Essl, G., Halderman, J. A., Mao, Z. M., & Prakash, A. (2014). Tivos: Trusted visual i/o paths for android. *University of Michigan CSE Technical Report CSE-TR-586-14*.

for Developers, A. (2022). *Permissions on android.* Retrieved from https://developer.android.com/guide/topics/permissions/overview (Retrieved February 20, 2022)

Frigo, P., Giuffrida, C., Bos, H., & Razavi, K. (2018). Grand pwning unit: Accelerating microarchitectural attacks with the gpu. In *2018 ieee symposium on security and privacy (sp)* (pp. 195–210).

Ghasempouri, T., Raik, J., Paul, K., Reinbrecht, C., Hamdioui, S., et al. (2021). Verifying cache architecture vulnerabilities using a formal security verification flow. *Microelectronics Reliability*, *119*, 114085.

Goel, D., & Jain, A. K. (2018). Mobile phishing attacks and defence mechanisms: State of art and open research challenges. *Computers & Security*, *73*, 519–544.

Golle, P., & Partridge, K. (2009). On the anonymity of home/work location pairs. In *International conference on pervasive computing* (pp. 390–397).

Google. (2022). *Google assistant.* Retrieved from https://assistant.google.com/ (Retrieved February 20, 2022)

Goucher, W. (2011). Look behind you: the dangers of shoulder surfing. *Computer Fraud & Security*, *2011*(11), 17–20.

Griswold-Steiner, I., LeFevre, Z., & Serwadda, A. (2021). Smartphone speech privacy concerns from side-channel attacks on facial biomechanics. *Computers & Security*, *100*, 102110.

Grünerbl, A., Muaremi, A., Osmani, V., Bahle, G., Oehler, S., Tröster, G., ... Lukowicz, P. (2014). Smartphone-based recognition of states and state changes in bipolar disorder patients. *IEEE journal of biomedical and health informatics*, *19*(1), 140–148.

Gruss, D., Bidner, D., & Mangard, S. (2015). Practical memory deduplication attacks in sandboxed javascript. In *European symposium on research in computer security* (pp. 108–122).

Gugenheimer, J., De Luca, A., Hess, H., Karg, S., Wolf, D., & Rukzio, E. (2015). Colorsnakes: Using colored decoys to secure authentication in sensitive contexts. In *Proceedings of the 17th international conference on human-computer interaction with mobile devices and services* (pp. 274–283).

Gulmezoglu, B., Zankl, A., Tol, M. C., Islam, S., Eisenbarth, T., & Sunar, B. (2019). Undermining user privacy on mobile devices using ai. In *Proceedings of the 2019 acm asia conference on computer and communications security* (pp. 214–227).

Guo, Y., Ma, J., Wu, W., & Chen, X. (2018). Inferring ui states of mobile applications through power side channel exploitation. In *International conference on security and privacy in communication systems* (pp. 210–226).

Gupta, H., Sural, S., Atluri, V., & Vaidya, J. (2016). Deciphering text from touchscreen key taps. In *Ifip annual conference on data and applications security and privacy* (pp. 3–18).

Gupta, S., Singhal, A., & Kapoor, A. (2016). A literature survey on social engineering attacks: Phishing attack. In *2016 international conference on computing, communication and automation (iccca)* (pp. 537–540).

Hanna, S., Huang, L., Wu, E., Li, S., Chen, C., & Song, D. (2012). Juxtapp: A scalable system for detecting code reuse among android applications. In *International conference on detection of intrusions and malware, and vulnerability assessment* (pp. 62–81).

Harbach, M., Von Zezschwitz, E., Fichtner, A., De Luca, A., & Smith, M. (2014). {It's} a hard lock life: A field study of smartphone ({Un) Locking} behavior and risk perception. In *10th symposium on usable privacy and security (soups 2014)* (pp. 213–230).

Heartfield, R., & Loukas, G. (2015). A taxonomy of attacks and a survey of defence mechanisms for semantic social engineering attacks. *ACM Computing Surveys (CSUR)*, *48*(3), 1–39.

Hernandez, J., McDuff, D. J., & Picard, R. W. (2015). Biophone: Physiology monitoring from peripheral smartphone motions. In *2015 37th annual international conference of the ieee engineering in medicine and biology society (embc)* (pp. 7180–7183).

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition* (Vol. 1, pp. 278–282).

Hölzl, M., Roland, M., & Mayrhofer, R. (2017). Real-world identification for an extensible and privacy-preserving mobile eid. In *Ifip international summer school on privacy and identity management* (pp. 354–370).

Hua, J., Shen, Z., & Zhong, S. (2016). We can track you if you take the metro: Tracking metro riders using accelerometers on smartphones. *IEEE Transactions on Information Forensics and Security*, *12*(2), 286–297.

Huestegge, L., & Pimenidis, L. (2014). Visual search in authentication systems based on memorized faces: Effects of memory load and retention interval. *International Journal of Human-Computer Interaction*, 604–611.

Inci, M. S., Eisenbarth, T., & Sunar, B. (2017). Hit by the bus: Qos degradation attack on android. In *Proceedings of the 2017 acm on asia conference on computer and communications security* (pp. 716–727).

Ivaturi, K., & Janczewski, L. (2011). A taxonomy for social engineering attacks. In *International conference on information resources management* (pp. 1–12).

Jana, S., & Shmatikov, V. (2012). Memento: Learning secrets from process footprints. In *2012 ieee symposium on security and privacy* (pp. 143–157).

Jia, J., & Gong, N. Z. (2018). {AttriGuard}: A practical defense against attribute inference attacks via adversarial machine learning. In *27th usenix security symposium (usenix security 18)* (pp. 513–529).

Jiang, L., Meng, W., Wang, Y., Su, C., & Li, J. (2017). Exploring energy consumption of juice filming charging attack on smartphones: a pilot study. In *International conference on network and system security* (pp. 199–213).

Johnson, A. (2021). Side channel attacks and mitigations 2015-2020: A taxonomy of published work. In *European conference on cyber warfare and security* (pp. 482–XII).

Joy Persial, G., Prabhu, M., & Shanmugalakshmi, R. (2011). Side channel attack-survey. *Int J Adva Sci Res Rev*, *1*(4), 54–57.

Kalal, Z., Mikolajczyk, K., & Matas, J. (2011). Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, *34*(7), 1409–1422.

Khamis, M., Alt, F., Hassib, M., von Zezschwitz, E., Hasholzner, R., & Bulling, A. (2016). Gazetouchpass: Multimodal authentication using gaze and touch on mobile devices. In *Proceedings of the 2016 chi conference extended abstracts on human factors in computing* *systems* (pp. 2156–2164).

Khamis, M., Eiband, M., Zürn, M., & Hussmann, H. (2018). Eyespot: Leveraging gaze to protect private text content on mobile devices from shoulder surfing. *Multimodal Technologies and Interaction*, *2*(3), 45.

Kirkwood, D., Tombul, C., Firth, C., Macdonald, F., Priftis, K., Mathis, F., ... Marky, K. (2022). Pin scrambler: Assessing the impact of randomized layouts on the usability and security of pins. In *Proceedings of the 21st international conference on mobile and ubiquitous multimedia* (pp. 83–88).

Krombholz, K., Hobel, H., Huber, M., & Weippl, E. (2015). Advanced social engineering attacks. *Journal of Information Security and applications*, *22*, 113–122.

Krombholz, K., Hupperich, T., & Holz, T. (2016). Use the force: Evaluating {Force-Sensitive} authentication for mobile devices. In *Twelfth symposium on usable privacy and security (soups 2016)* (pp. 207–219).

Ku, Y., Park, L. H., Shin, S., & Kwon, T. (2019). Draw it as shown: Behavioral pattern lock for mobile user authentication. *IEEE Access*, *7*, 69363–69378.

Kumar, R., Kishore, S., Lu, H., & Prakash, A. (2020). Security analysis of unified payments interface and payment apps in india. In *29th usenix security symposium (usenix security 20)* (pp. 1499–1516).

Lakshmanan, N., Budhdev, N., Kang, M. S., Chan, M. C., & Han, J. (2021). A stealthy location identification attack exploiting carrier aggregation in cellular networks. In *30th usenix security symposium (usenix security 21)* (pp. 3899–3916).

La Polla, M., Martinelli, F., & Sgandurra, D. (2012). A survey on security for mobile devices. *IEEE communications surveys & tutorials*, *15*(1), 446–471.

Lee, K., Kaiser, B., Mayer, J., & Narayanan, A. (2020). An empirical study of wireless carrier authentication for {SIM} swaps. In *Sixteenth symposium on usable privacy and security (soups 2020)* (pp. 61–79).

Li, M., Meng, Y., Liu, J., Zhu, H., Liang, X., Liu, Y., & Ruan, N. (2016). When csi meets public wifi: inferring your mobile phone password via wifi signals. In *Proceedings of the 2016 acm sigsac conference on computer and communications security* (pp. 1068–1079).

Li, Z., Ma, F., Rathore, A. S., Yang, Z., Chen, B., Su, L., & Xu, W. (2020). Wavespy: Remote and through-wall screen attack via mmwave sensing. In *2020 ieee symposium on security and privacy (sp)* (pp. 217–232).

LiKamWa, R., Liu, Y., Lane, N. D., & Zhong, L. (2013). Moodscope: Building a mood sensor from smartphone usage patterns. In *Proceeding of the 11th annual international conference on mobile systems, applications, and services* (pp. 389–402).

Lipp, M., Gruss, D., Spreitzer, R., Maurice, C., & Mangard, S. (2016). {ARMageddon}: Cache attacks on mobile devices. In *25th usenix security symposium (usenix security 16)* (pp. 549–564).

Liu, P., Zang, W., & Yu, M. (2005). Incentive-based modeling and inference of attacker intent, objectives, and strategies. *ACM Transactions on Information and System Security (TISSEC)*, *8*(1), 78–118.

Luo, T., Jin, X., Ananthanarayanan, A., & Du, W. (2012). Touchjacking attacks on web in android, ios, and windows phone. In *International symposium on foundations and practice of security* (pp. 227–243).

Ma, J., Yang, W., Luo, M., & Li, N. (2014). A study of probabilistic password models. In *2014 ieee symposium on security and privacy* (pp. 689–704).

Marky, K., Macdonald, S., Abdrabou, Y., & Khamis, M. (2023). In the quest to protect users from {Side-Channel} attacks–a {User-Centred} design space to mitigate thermal attacks on public payment terminals. In *32nd usenix security symposium (usenix security 23)* (pp. 5235–5252).

Matas, J., Galambos, C., & Kittler, J. (2000). Robust detection of lines using the progressive probabilistic hough transform. *Computer vision and image understanding*, *78*(1), 119–137.

Matyunin, N., Wang, Y., Arul, T., Kullmann, K., Szefer, J., & Katzenbeisser, S. (2019). Magneticspy: Exploiting magnetometer in mobile devices for website and application fingerprinting. In *Proceedings of the 18th acm workshop on privacy in the electronic society* (pp. 135–149).

Mayrhofer, R., & Sigg, S. (2021). Adversary models for mobile device authentication. *ACM Computing Surveys (CSUR)*, *54*(9), 1–35.

Mayring, P., et al. (2004). Qualitative content analysis. *A companion to qualitative research*, *1*(2), 159–176.

Meng, W., Lee, W. H., Murali, S., & Krishnan, S. (2015). Charging me and i know your secrets! towards juice filming attacks on smartphones. In *Proceedings of the 1st acm workshop on cyber-physical system security* (pp. 89–98).

Mitchell, M., Wang, A.-I., & Reiher, P. (2015). Cashtags: Prevent leaking sensitive information through screen display. In *Proceedings of the usenix security symposium* (Vol. 1).

Myagmar, S., Lee, A. J., & Yurcik, W. (2005). Threat modeling as a basis for security requirements. In *Symposium on requirements engineering for information security (sreis)* (Vol. 2005, pp. 1–8).

Narain, S., Sanatinia, A., & Noubir, G. (2014). Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning. In *Proceedings of the 2014 acm conference on security and privacy in wireless & mobile networks* (pp. 201–212).

Narain, S., Vo-Huu, T. D., Block, K., & Noubir, G. (2016). Inferring user routes and locations using zero-permission mobile sensors. In *2016 ieee symposium on security and privacy (sp)* (pp. 397–413).

Niemietz, M., & Schwenk, J. (2012). Ui redressing attacks on android devices. *Black Hat Abu Dhabi*.

OCR, T. (2022). *Tesseract ocr.* Retrieved from https://github.com/tesseract-ocr/tesseract (Retrieved February 20, 2022)

Ometov, A., Levina, A., Borisenko, P., Mostovoy, R., Orsino, A., & Andreev, S. (2017). Mobile social networking under side-channel attacks: Practical security challenges. *IEEE Access*, *5*, 2591–2601.

OpenCV. (2022a). *Eroding and dilating.* Retrieved from https://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html (Retrieved February 20, 2022)

OpenCV. (2022b). *Opencv.* Retrieved from https://opencv.org (Retrieved February 20, 2022)

OpenCV. (2022c). *Opencv: Canny edge detection.* Retrieved from https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html (Retrieved February 20, 2022)

Owusu, E., Han, J., Das, S., Perrig, A., & Zhang, J. (2012). Accessory: password inference using accelerometers on smartphones. In *proceedings of the twelfth workshop on mobile computing systems & applications* (pp. 1–6).

Pham, A., Dacosta, I., Losiouk, E., Stephan, J., Huguenin, K., & Hubaux, J.-P. (2019). {HideMyApp}: Hiding the presence of sensitive apps on android. In *28th usenix security symposium (usenix security 19)* (pp. 711–728).

Ping, D., Sun, X., & Mao, B. (2015). Textlogger: inferring longer inputs on touch screen using motion sensors. In *Proceedings of the 8th acm conference on security & privacy in wireless and mobile networks* (pp. 1–12).

Por, L. Y., Ng, I. O., Chen, Y.-L., Yang, J., & Ku, C. S. (2024). A systematic literature review on the security attacks and countermeasures used in graphical passwords. *IEEE Access*.

Possemato, A., Lanzi, A., Chung, S. P. H., Lee, W., & Fratantonio, Y. (2018). Clickshield: Are you hiding something? towards eradicating clickjacking on android. In *Proceedings of the 2018 acm sigsac conference on computer and communications security* (pp. 1120–1136).

Raguram, R., White, A. M., Goswami, D., Monrose, F., & Frahm, J.-M. (2011). ispy: automatic reconstruction of typed input from compromising reflections. In *Proceedings of the 18th acm conference on computer and communications security* (pp. 527–536).

Rescorla, E., & Korver, B. (2003). *Guidelines for writing rfc text on security considerations* (Tech. Rep.). BCP 72, RFC 3552, July.

SAAD, A., LIEBERS, J., GRUENEFELD, U., ALT, F., & SCHNEEGASS, S. (2021). Understanding bystanders' tendency to shoulder surf smartphones using 360-degree videos in virtual reality. , 1–8.

Security, H. (2022). *Android permissions can be dangerous: Full guide to managing them.* Retrieved from https://heimdalsecurity.com/blog/android-permissions-full-guide/ (Retrieved February 20, 2022)

Security, N. E. (2022). *Uniqpass v15 – large password list.* Retrieved from https://neverendingsecurity.wordpress.com/2015/04/19/uniqpass-v15-large-password-list/ (Retrieved February 20, 2022)

Shi, E., Niu, Y., Jakobsson, M., & Chow, R. (2010). Implicit authentication through learning user behavior. In *International conference on information security* (pp. 99–113).

Shin, H., Sim, S., Kwon, H., Hwang, S., & Lee, Y. (2022). A new smart smudge attack using cnn. *International Journal of Information Security*, *21*(1), 25–36.

Shukla, D., & Phoha, V. V. (2019). Stealing passwords by observing hands movement. *IEEE Transactions on Information Forensics and Security*, *14*(12), 3086–3101.

Simon, L., & Anderson, R. (2013). Pin skimmer: inferring pins through the camera and microphone. In *Proceedings of the third acm workshop on security and privacy in smartphones & mobile devices* (pp. 67–78).

Simon, L., Xu, W., & Anderson, R. (2016). Don't interrupt me while i type: Inferring text entered through gesture typing on android keyboards.

Son, S., Kim, D., & Shmatikov, V. (2016). What mobile ads know about mobile users. In *Ndss.*

Song, C., Lin, F., Ba, Z., Ren, K., Zhou, C., & Xu, W. (2016). My smartphone knows what you print: Exploring smartphone-based side-channel attacks against 3d printers. In *Proceedings of the 2016 acm sigsac conference on computer and communications security* (pp. 895–907).

Spreitzer, R. (2014). Pin skimming: exploiting the ambient-light sensor in mobile devices. In *Proceedings of the 4th acm workshop on security and privacy in smartphones & mobile devices* (pp. 51–62).

Spreitzer, R., Griesmayr, S., Korak, T., & Mangard, S. (2016). Exploiting data-usage statistics for website fingerprinting attacks on android. In *Proceedings of the 9th acm conference on security & privacy in wireless and mobile networks* (pp. 49–60).

Spreitzer, R., Kirchengast, F., Gruss, D., & Mangard, S. (2018). Procharvester: Fully automated analysis of procfs side-channel leaks on android. In *Proceedings of the 2018 on asia conference on computer and communications security* (pp. 749–763).

Spreitzer, R., Palfinger, G., & Mangard, S. (2018). Scandroid: Automated side-channel analysis of android apis. In *Proceedings of the 11th acm conference on security & privacy in wireless and mobile networks* (pp. 224–235).

srsRAN. (2022). *srsran - your own mobile network.* Retrieved from https://www.srslte.com (Retrieved February 20, 2022)

Tang, B., Wang, Z., Wang, R., Zhao, L., & Wang, L. (2018). Niffler: A context-aware and user-independent side-channel attack system for password inference. *Wireless Communications and Mobile Computing*, *2018*.

Tsalis, N., Vasilellis, E., Mentzelioti, D., & Apostolopoulos, T. (2019). A taxonomy of side channel attacks on critical infrastructures and relevant systems. In *Critical infrastructure security and resilience* (pp. 283–313). Springer.

Ulqinaku, E., Malisa, L., Stefa, J., Mei, A., & Čapkun, S. (2017). Using hover to compromise the confidentiality of user input on android. In *Proceedings of the 10th acm conference on security and privacy in wireless and mobile networks* (pp. 12–22).

Von Gioi, R. G., Jakubowicz, J., Morel, J.-M., & Randall, G. (2008). Lsd: A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, *32*(4), 722–732.

Wang, J., Gao, B., Tu, H., Liang, H.-N., Liu, Z., Luo, W., & Weng, J. (2023). Secure and memorable authentication using dynamic combinations of 3d objects in virtual reality. *International Journal of Human–Computer Interaction*, 1–19.

Watanabe, T., Akiyama, M., & Mori, T. (2015). {RouteDetector}: Sensor-based positioning system that exploits {Spatio-Temporal} regularity of human mobility. In *9th usenix workshop on offensive technologies (woot 15).*

Wu, C., He, K., Chen, J., & Du, R. (2019). Icauth: Implicit and continuous authentication when the screen is awake. In *Icc 2019-2019 ieee international conference on communications (icc)* (pp. 1–6).

Yan, L., Guo, Y., Chen, X., & Mei, H. (2015). A study on power side channels on mobile devices. In *Proceedings of the 7th asia-pacific symposium on internetware* (pp. 30–38).

Yang, L., Zhi, Y., Wei, T., Yu, S., & Ma, J. (2019). Inference attack in android activity based on program fingerprint. *Journal of Network and Computer Applications*, *127*, 92–106.

Ye, G., Tang, Z., Fang, D., Chen, X., Kim, K. I., Taylor, B., & Wang, Z. (2017). Cracking android pattern lock in five attempts. In *Proceedings of the 2017 network and distributed system security symposium 2017 (ndss 17).*

Yu, X., Wang, Z., Li, Y., Li, L., Zhu, W. T., & Song, L. (2017). Evopass: Evolvable graphical password against shoulder-surfing attacks. *Computers & Security*, *70*, 179–198.

Yue, Q., Ling, Z., Fu, X., Liu, B., Ren, K., & Zhao, W. (2014). Blind recognition of touched keys on mobile devices. In *Proceedings of the 2014 acm sigsac conference on computer and communications security* (pp. 1403–1414).

Zhang, J., Tang, Z., Li, M., Fang, D., Chen, X., & Wang, Z. (2019). Find me a safe zone: A countermeasure for channel state information based attacks. *Computers & Security*, *80*, 273–290.

Zhang, J., Zheng, X., Tang, Z., Xing, T., Chen, X., Fang, D., . . . Chen, F. (2016). Privacy leakage in mobile sensing: Your unlock passwords can be leaked through wireless hotspot functionality. *Mobile Information Systems*, *2016*.

Zhang, N., Sun, K., Shands, D., Lou, W., & Hou, Y. T. (2016). Truspy: Cache side-channel information leakage from the secure world on arm devices. *Cryptology ePrint Archive*.

Zhang, X., Wang, X., Bai, X., Zhang, Y., & Wang, X. (2018). Os-level side channels without procfs: Exploring cross-app information leakage on ios. In *Proceedings of the symposium on network and distributed system security.*

Zhang, X., Xiao, Y., & Zhang, Y. (2016). Return-oriented flush-reload side channels on arm and their implications for android devices. In *Proceedings of the 2016 acm sigsac conference on computer and communications security* (pp. 858–870).

Zheng, H., & Hu, H. (2019). Missile: A system of mobile inertial sensor-based sensitive indoor location eavesdropping. *IEEE Transactions on Information Forensics and Security*, *15*, 3137–3151.

Zhou, W., Zhang, X., & Jiang, X. (2013). Appink: watermarking android apps for repackaging deterrence. In *Proceedings of the 8th acm sigsac symposium on information, computer and communications security* (pp. 1–12).

Zhou, W., Zhou, Y., Jiang, X., & Ning, P. (2012). Detecting repackaged smartphone applications in third-party android marketplaces. In *Proceedings of the second acm conference on data and application security and privacy* (pp. 317–326).

Zisserman, A. (2015). *The svm classifier.* Retrieved from https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf (Retrieved February 20, 2022)

Appendix A. Papers Included in the Systematic Literature Review

Appendix B. Codebook Used for Coding the Attack Requirements

Table A1. The Table shows the list of papers extracted from the selected publication venues which are included in the systematic literature review.

Papers Included in the Categorisation (I)	**Reference**
Main Search List	
Undermining User Privacy on Mobile Devices Using AI	(Gulmezoglu et al., 2019)
Charging Me and I Know Your Secrets!: Towards Juice Filming Attacks on Smartphones	(Meng et al., 2015)
Boosting the Guessing Attack Performance on Android Lock Patterns with Smudge Attacks	(Cha et al., 2017)
Find me a safe zone: A countermeasure for channel state information based attacks	(J. Zhang et al., 2019)
EvoPass: Evolvable graphical password against shoulder-surfing attacks	(Yu et al., 2017)
Smartphone speech privacy concerns from side-channel attacks on facial biomechanics	(Griswold-Steiner et al., 2021)
Inferring User Routes and Locations Using Zero-Permission Mobile Sensors	(Narain et al., 2016)
MISSILE: A System of Mobile Inertial Sensor-Based Sensitive Indoor Location Eavesdropping	(Zheng & Hu, 2019)
Stealing Passwords by Observing Hands Movement	(Shukla & Phoha, 2019)
We Can Track You if You Take the Metro: Tracking Metro Riders Using Accelerometers on Smartphones	(Hua et al., 2016)
Peeking into your app without actually seeing it:{UI} state inference and novel android attacks	(Q. A. Chen et al., 2014)
Armageddon: Cache attacks on mobile devices	(Lipp et al., 2016)
Security analysis of Unified Payments Interface and payment apps in India	(Kumar et al., 2020)
A Stealthy Location Identification Attack Exploiting Carrier Aggregation in Cellular Networks	(Lakshmanan et al., 2021)
Cashtags: Prevent leaking sensitive information through screen display	(Mitchell et al., 2015)
A closer look at recognition-based graphical passwords on mobile devices	(Dunphy et al., 2010)
An empirical study of wireless carrier authentication for SIM swaps	(Lee et al., 2020)
Hit by the Bus: QoS Degradation Attack on Android	(Inci et al., 2017)
ProcHarvester: Fully Automated Analysis of Procfs Side-Channel Leaks on Android	(Spreitzer, Kirchengast, et al., 2018)
WaveSpy: Remote and Through-wall Screen Attack via mmWave Sensing	(Z. Li et al., 2020)

Table A2. The Table shows the list of papers extracted by performing Backward Search which are included in the systematic literature review.

Papers Included in the Categorisation (II)	**Reference**
Backward Search List	
Practical memory deduplication attacks in sandboxed javascript	(Gruss et al., 2015)
Memento: Learning secrets from process footprints	(Jana & Shmatikov, 2012)
Scandroid: Automated side-channel analysis of android apis	(Spreitzer, Palfinger, & Mangard, 2018)
Os-level side channels without procfs: Exploring cross-app information leakage on ios	(X. Zhang et al., 2018)
Accessory: password inference using accelerometers on smartphones	(Owusu et al., 2012)
A pilot study on the security of pattern screen-lock methods and soft side channel attacks	(Andriotis et al., 2013)
When CSI meets public wifi: Inferring your mobile phone password via wifi signals	(M. Li et al., 2016)
Cracking android pattern lock in five attempts	(Ye et al., 2017)
Blind recognition of touched keys on mobile devices	(Yue et al., 2014)
Privacy leakage in mobile sensing: your unlock passwords can be leaked through wireless hotspot functionality	(J. Zhang et al., 2016)
Routedetector: Sensor-based positioning system that exploits spatio-temporal regularity of human mobility	(Watanabe et al., 2015)
Mobile social networking under side-channel attacks: Practical security challenges	(Ometov et al., 2017)
Smartphone passcode prediction	(T. Chen et al., 2018)
iSpy: Automatic reconstruction of typed input from compromising reflections	(Raguram et al., 2011)
Pin skimmer: Inferring pins through the camera and microphone	(Simon & Anderson, 2013)
Niffler: A contextaware and user-independent side-channel attack system for password inference	(Tang et al., 2018)
Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning	(Narain et al., 2014)
PIN Skimming: Exploiting the Ambient-Light Sensor in Mobile Devices	(Spreitzer, 2014)
What the App is That? Deception and Countermeasures in the Android User Interface	(Bianchi et al., 2015)
Don't Interrupt Me While I Type: Inferring Text Entered Through Gesture Typing on Android Keyboards	(Simon et al., 2016)
Exploiting Data-Usage Statistics for Website Fingerprinting Attacks on Android	(Spreitzer et al., 2016)
A Study on Power Side Channels on Mobile Devices	(Yan et al., 2015)
Return-Oriented Flush- Reload Side Channels on ARM and Their Implications for Android Devices	(X. Zhang et al., 2016)

Table A3. The Table shows the list of papers extracted by performing Forward Search which are included in the systematic literature review.

Papers Included in the Categorisation (III)	**Reference**
Forward Search List	
Deciphering text from touchscreen key taps	(H. Gupta et al., 2016)
Exploring energy consumption of juice filming charging attack on smartphones: a pilot study	(Jiang et al., 2017)
Draw it as shown: Behavioral pattern lock for mobile user authentication	(Ku et al., 2019)
Syspal: System-guided pattern locks for android	(Cho et al., 2017)
A new smart smudge attack using CNN	(Shin et al., 2022)
Inference attack in android activity based on program fingerprint	(Yang et al., 2019)
Inferring UI States of Mobile Applications Through Power Side Channel Exploitation	(Guo et al., 2018)
No pardon for the interruption: New inference attacks on android through interrupt timing analysis	(Diao et al., 2016)
MagneticSpy: Exploiting Magnetometer in Mobile Devices for Website and Application Fingerprinting	(Matyunin et al., 2019)
Using hover to compromise the confidentiality of user input on Android	(Ulqinaku et al., 2017)
Textlogger: inferring longer inputs on touch screen using motion sensors	(Ping et al., 2015)
Clickshield: Are you hiding something? Towards eradicating clickjacking on Android	(Possemato et al., 2018)
Hidemyapp: Hiding the presence of sensitive apps on android	(Pham et al., 2019)
Gui-squatting attack: Automated generation of android phishing apps	(S. Chen et al., 2019)
{AttriGuard}: A practical defense against attribute inference attacks via adversarial machine learning	(Jia & Gong, 2018)
Resource Race Attacks on Android	(Cai et al., 2020)
What Mobile Ads Know About Mobile Users.	(Son et al., 2016)
Grand pwning unit: Accelerating microarchitectural attacks with the GPU	(Frigo et al., 2018)
Truspy: Cache side-channel information leakage from the secure world on arm devices	(N. Zhang et al., 2016)
Schrodintext: Strong protection of sensitive textual content of mobile applications	(Amiri Sani, 2017)
ICAUTH: Implicit and continuous authentication when the screen is awake	(Wu et al., 2019)
Tivos: Trusted visual i/o paths for android	(Fernandes et al., 2014)

Table B1. The Table shows the codebook for Attack Infrastructure Requirement Categories

Category	Description	Examples
Manual Tools	Refers to non-electronic/non-powered devices or tools	Pen, box, papers, sealed box
Software Tools	programs that make use of sophisticated algorithms	Smug attack tool, n-gram Markov Model, image matching algorithm, Probabilistic Hough Transformation, Android Background Service, CSI Measurement Tool, Voice Training Data from the accomplice, VGA2USB driver, Remote Server, Probabilistic Password Model (eg n-gram Markov Model), Supervised Machine Learning Model, Model Classifier Configurations, Android Framework Services, Edge Detection Algorithm, Edge dilation, CV algorithm, open-source cellular projects. Edge Detection Algorithm, Tracking Learning Detection, Dynamic Time Wrapping, Video Editing Tool, Fine-Grained Accelerometer Data, Keypress segmentation, Probabilistic Keypress Classification, Probabilistic Error Model, Search Algorithm, Graph construction, Android NDK, ADB, CSI Measurement Tool, discrete wavelet decomposition, Threshold Quantification, Random Forest, Skin Detection Algorithm, Inverse Wavelet Transform, Dynamic Time Warping, AdSDK, Symbolic Aggregate approXimation (SAX), LibSVM, exotic atomic operation loop, cache profiling tool, sticky background service, LibSVM, malloc implementation (GNU C Library), signal processing scheme, wavelet-based response analysis, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), standard Android Framework services, alternate soft-keyboards, deformable part-based model (DPM), k-means clustering algorithm, APK tool, Homography, Keras, ADB shell, Tensorflow, AdSDK.
Mobile Phone Application	An application that needs to be installed on the target's mobile device	Malicious application, legitimate spyware, privileged application, Trojan application, phishing application, malware, non-malicious application
Advanced Programming	Advanced programming expertise from specialized fields of programming	Image Processing, Perspective Transform Technique, Canny Edge Detection, Hough Circle Transform, Deep Neural Network, C++/Java, Hidden Markov Model, Machine Learning, Supervised Learning Scheme, Genetic and Detection Algorithm, Pattern Matching Algorithm, Recurrent Neural Network, Neural Network Processes, Deep Learning, Weka Toolkit, Convolutional Neural Network, Python, OCR Techniques, Matlab LTE Toolkit, Computer Vision, Skin Segmentation Techniques, finger detection classifier, Supervised Learning Scheme, Support Vector Machine, OpenCV, Classifier, Language Model, regression model, classifier, Javascript, C/C++, genetic and detection algorithm, Gaussian filter, pattern matching algorithm, Recurrent neural network (RNN), Java/C language, Java-ML Library, matlab LTE toolbox, cascade classifier training, Matlab's Statistics Toolbox, sandbox app, Symlet Filter, natural language processing algorithms, signal processing techniques, scikit-learn library, kernel privileges, Return-Oriented Programming, code injection
Hardware Tools	External electronic tools required to be connected with the attack setup	VGA/USB interface, Micro USB connector, Mobile High Definition Link (MHL) standard, computer, Rasberry Pi, High-resolution camera, flash lightning system, wireless router, video recorder, Bluetooth Low Energy Beacon (BLE), Low Power Microcontroller, EspressifESP32 chip, a dual-core Tensilica Extensa LX6 processor, High Frequency Analog to Digital Converter, smartphone, USB outlet, charging cable, power bank, voltage monitor, SD card, software-defined radio device, camcorder, surveillance camera, public geographical data, a similar device as victim's, FMCW mmWave probe, frequency-modulated continuous-wave (FMCW) radar, Panasonic Lumix DMC-TZ5 compact camera, Gorilla Glass screen, USB microscope with 400x magnification, FLIR E30, hard light source, Digital Single-Lens Reflex (DSLR), laptop with Intel 5300 NIC, smart device with hotspot functionality, external sound card,Monsoon Power Monitor , smart device with hotspot functionality, Freescale i.MX53 development board running CortexA-8 processor
User Phone Permissions	Permissions requested by the attacker to access different services or sensors on the mobile device	Access to camera, microphone, accelerometer sensor, orientation sensor, internet, external storage, get_tasks, system_alert, gyroscope, magnetometer, Bluetooth, WRITE_EXTERNAL_STATE, Receive SMS, Read Phone State, GET_TASKS permissions, motion sensor
Human Capabilities	Resources within the scope of human physical and personal abilities	Close proximity with the target, direct observation, the human memory. physical access to the target device, physical walk through the target's location, context information about the victim, access to public geographical information, access to a reference image of the phone, Victim's phone number and name,