2019 IEEE Symposium on Security and Privacy

Tap 'n Ghost: A Compilation of Novel Attack Techniques against Smartphone Touchscreens

Seita Maruyama
Waseda University, Japan
maruyama@nsl.cs.waseda.ac.jp

Satohiro Wakabayashi
Waseda University, Japan
wakabayashi@goto.info.waseda.ac.jp

Tatsuya Mori
Waseda University / RIKEN AIP, Japan
mori@nsl.cs.waseda.ac.jp

***Abstract*—We present a novel attack named "*Tap 'n Ghost*", which aims to attack the touchscreens of NFC-enabled mobile devices such as smartphones. *Tap 'n Ghost* consists of two striking attack techniques—"*Tag-based Adaptive Ploy* (*TAP*)" and "*Ghost Touch Generator*." First, using an NFC card emulator embedded in a common object such as table, a *TAP* system performs tailored attacks on the victim's smartphone by employing device fingerprinting; e.g., popping up a customized dialog box asking whether or not to connect to an attacker's Bluetooth mouse. Further, *Ghost Touch Generator* forces the victim to connect to the mouse even if she or he aimed to cancel the dialog by touching the "cancel" button; i.e., it alters the selection of a button on a screen. After the connection is established, the attacker can remotely take control of the smartphone, with the knowledge about the layout of the screen derived from the device fingerprinting. To evaluate the practicality of the attack, we perform an online survey with 300 respondents and a user study involving 16 participants. The results demonstrate that the attack is realistic. We additionally discuss the possible countermeasures against the threats posed by *Tap 'n Ghost*.**

I. Introduction

Today, we use a smartphone not only for accessing the various Internet services but also for interacting with the networked devices around us, e.g., wireless headphones, fitness devices, smart home devices, connected cars, and contactless payment systems. To communicate with these networked devices, modern smartphones are shipped with various networking interfaces such as cellular networks, Wi-Fi, Bluetooth, and NFC. This trend has helped smartphones to get more and more connected to our life — anywhere, anytime and with anything.

Given the pervasive network connectivity of smartphones, we propose a novel proof-of-concept attack that is named "*Tap 'n Ghost*." The *Tap 'n Ghost* attack is aimed at targeting the touchscreens of NFC-enabled mobile devices such as smartphones. The *Tap 'n Ghost* consists of two striking techniques—"*Tag-based Adaptive Ploy* (*TAP*)" and "*Ghost Touch Generator*." These techniques enable an attacker to trigger malicious events on the victim's smartphone, such as connecting to an attacker's Bluetooth mouse, and allows the victim to connect to it despite the fact that she or he intended to cancel the event by touching the "cancel" button. After the connection is established, the attacker can remotely take control of the smartphone. These techniques can be covertly embedded into common objects such as a table. As the common objects are routinely encountered in daily life, they will never be considered as an NFC touchpoint. In the following, we briefly describe the two techniques.

TAP consists of a processor, a communication interface, such as a Wi-Fi interface, and an NFC-tag emulator, which is a device that makes use of the NFC card emulation mode and can act as multiple NFC tags; thus, it can generate *tailored* attacks on a targeted smartphone. The standard operation of *TAP* is as follows. First, it works as an NFC tag, which is programmed to visit a malicious URL. A victim device that read the tag will be redirected to the URL. Further, *TAP* works with a web server behind the URL. The web server fingerprints the victim device and conveys the type of the device to *TAP*. Next, *TAP* uses this information to tailor additional tags to be read by the victim device.

Although *TAP* can induce a victim device to perform certain low-risk actions, such as opening a URL without prompting the user, high-risk actions, such as pairing with a Bluetooth device, do require user confirmation. To deal with this problem, we develop a new technique named *Ghost Touch Generator*, which is aimed at deceiving victim devices into sensing "ghost touch events" on their touch screens. The attack can alter the user's selection; i.e., even though a victim touches a "CANCEL" button, the attack can make the operating system recognize the event as a touch of another button, "CONNECT." The key idea underlying *Ghost Touch Generator* is to cause an intentional malfunction by injecting intentional noise signals externally. We found that producing large alternating voltages at a specific frequency near a touchscreen can cause a malfunction due to capacitive coupling with the RX electrodes.

To the best of our knowledge, our work is the first to study the threat of active attack against touchscreens; that is, intentionally radiating signals toward a touchscreen to cause targeted malfunctions. On the other hand, our work is not the first to study the threat of maliciously programmed NFC tags. There have been several works that have addressed the issue [1]–[6]. An attacker can leverage an NFC tag to trigger risky actions; e.g., opening a malicious URL in a browser without user approval [4] or forcing a smartphone to pair with a rogue Bluetooth device [3], [4]. What distinguishes our approach from prior work is that by leveraging NFC card emulation, ours can achieve complex and tailored attacks, such as adjusting the attack parameters based on the victim's smartphone model; i.e., it can perform device fingerprinting. In addition, unlike the previously reported malicious NFC

DOI 10.1109/SP.2019.00037

attacks, our attack is *intangible*; instead of actively prompting victims to touch a point such as a smart poster where a malicious NFC tag is embedded, *Tap 'n Ghost* passively waits for victims to approach a malicious NFC tag emulator that is embedded within ordinary objects. Even if the NFC tag launches an event, victims will not realize that their devices are engaging in the NFC communication originated from such an ordinary object.

We make the following contributions:

- We present a novel class of attacks that we call *Tap 'n Ghost*, which injects malicious functionalities into common objects (Section III).
- We develop two effective techniques called "TAP" (Section IV) and "*Ghost Touch Generator*" (Section V).
- We demonstrate the technical feasibility of *Tap 'n Ghost* attacks using 24 smartphones for *TAP* experiments and 7 smartphones for *Ghost Touch Generator* experiments (Section VI).
- We demonstrate the practicality of the threat through an online survey with 300 respondents and an empirical user study with 16 participants (Section VII).
- We provide possible countermeasures against the threats of *Tap 'n Ghost* attacks (Section VIII).

To visually demonstrate the feasibility of our attack, we provide demo videos of the experiments. They are available at https://goo.gl/xoVt23.

II. Background

In this section, we provide background information on the two key technologies used in our attack, NFC and capacitive touchscreen, which are widely used in smartphones.

A. NFC

Near-Field Communication (NFC) is a short-range wireless communication technology widely used in many applications, e.g., contactless payment systems, transit passes, smart posters, and smartphone apps. While the theoretical working distance of NFC is up to 20 cm, the practical working distance is a maximum of about 4 cm. According to Ref. [7], the number of smartphones equipped with NFC is drastically increasing year by year. Roughly two-thirds of all smartphones shipped in 2018 are expected to be equipped with NFC.

NFC is a communication protocol that can exchange data just by bringing NFC compatible devices close to each other. In many NFC applications, communication is established without going through user interaction. This design leads to high usability. However, the high usability of NFC raises several security issues. Although the NFC communication range is limited to only a few centimeters and tags can be configured to be read-only, the NFC service can be easily exploited by a simple attack—replacing the existing NFC tag with a malicious NFC tag. Several studies have reported the threats of malicious NFC tags [1]–[6]. We will summarize these studies in Section IX.

TABLE I
Android OS operations that can be launched by reading an NFC tag.

operation	requests user approval
open a specified URL	No
launch a specified app	No
send an Intent to an NFC-enabled app	No
launch an Instant app	No
send email to specified address with specified subject and body	Yes
connect to specified Wi-Fi AP	Yes
pair with specified Bluetooth device	Yes

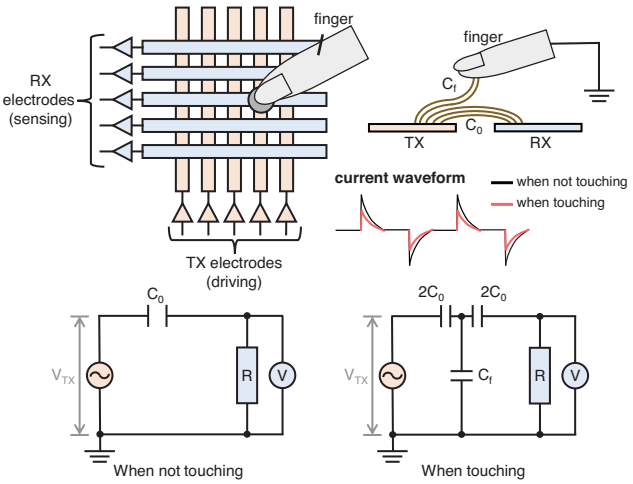

Fig. 1. Touch detection mechanism of mutual capacitance touchscreen [10].

Our attack exploits the NFC implementation of the Android OS version 4.1 or later[1]. According to the statistics shown in Ref. [8], the fraction of Android devices with OS versions older than 4.1 account for less than 1% as of October 2018. Thus, our NFC-based attack is applicable to most of the Android smartphones. When an Android device is held over an NFC tag, Android OS can perform various operations by reading the data recorded in the NFC tag. Table I lists the operations that can be launched by reading an NFC tag.

B. Capacitive Touchscreen

The majority of the current mobile devices, such as smartphones and tablets, are equipped with touchscreens. While there are various technologies for sensing touch, mutual capacitive sensors are widely used in smartphones as they have high resolution and multi-touch support [9].

As shown in Figure 1, a mutual capacitive touchscreen controller consists of the grid of the transmitter (TX) electrodes and receiver (RX) electrodes, which are mutually coupled, e.g., C_0 in the figure. These TX/RX electrodes are used for sensing touch events. As the human body has a capacitance, it

[1]Our attack exploits the following features: NFC Reader mode (supported from Android 2.3), Android Application Records (supported from Android 4.0), and Bluetooth pairing via NFC (supported from Android 4.1).

can act as a capacitor. When a finger approaches the screen's surface, the capacitance between the TX and RX electrodes will change. This is because the finger extracts an electric charge from the touchscreen through mutual capacitance (C_f in the figure). The changes in capacitance between the TX and RX electrodes, δC, will cause changes in electric charge, δQ, which is expressed as $\delta Q = \delta C \times V_{TX}$, where V_{TX} is TX driver voltage. Thus, the touchscreen controller can detect touches by measuring the changes in the amount of electric current that flows into the RX electrodes. The pair of TX and RX electrodes for which the changes are detected is used to locate the area of touch.

III. Overview of the Attack

In this section, we present the overview of *Tap 'n Ghost* attack. We first describe our threat model. We then present how the *TAP* attack can trigger high risk actions. Finally, we describe the end-to-end attack scenario and its consequences.

A. Threat model

In this work, we assume an attacker has embedded several components used for the attack, e.g., an NFC-tag emulator, a single-board computer, and high-voltage transformer, in a common object such as table. The cost of implementing such a system will be discussed in Sec. VII-E. We also assume the victim has an Android smartphone equipped with NFC. The victim unintentionally places the smartphone close to a *Tap 'n Ghost*-installed table, and the smartphone automatically reads a malicious NFC tag/emulator when it is unlocked and not in the sleep mode. In Section VII, we will test the validity of these assumptions through an online survey and an user study.

After reading the NFC Data Exchange Format (NDEF) records stored in the malicious NFC tag/emulator, the smartphone will execute a corresponding operation used for attacks, which will be described in the next subsection. As triggering high risk actions such as connecting to Wi-Fi AP requires user approval by displaying a dialog box with a confirmation message, we use two approaches to evade the user approval process. The first is to mislead the user into approving the dialog box by different ways of manipulating the UI such as showing a misleading message (Section IV). The second approach is an attack on the touchscreen named *Ghost Touch Generator* (Section V). Figure 2 summarizes the overview of the *Tap 'n Ghost* attack.

B. Triggering High Risk Actions

As shown in Table I, two types of operations can be invoked via NFC: operations that require user approval and operations that do not. The latter requirement will be automatically executed if an NFC tag is brought close to a smartphone. We call an attack that makes use of such operations as a *single-shot attack*. A representative example of a single-shot attack is opening a malicious URL in a browser; such a malicious website can trigger download/installation of a malware on the smartphone [5].

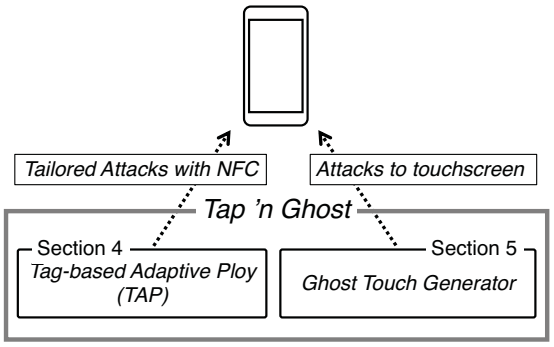

Fig. 2. Overview of the *Tap 'n Ghost* attack.

By combining multiple single-shot attacks, we can create more sophisticated attacks, which we call *Tag-based Adaptive Ploy* (*TAP*). *TAP* enables an attacker to establish various attacks including device fingerprinting. As shown in Section IV, device fingerprinting is useful to infer the language used for the device; the information can be used to display a dialog box with a misleading message to the victim. The fingerprint information can also be used for displaying a dialog box with a suitable message, which needs to be adaptive to the vendor-specific customization of confirmation message strings. Furthermore, the fingerprint information is necessary to optimize the success rate of the *Ghost Touch Generator*, which requires to adjust the frequency of external signal to be injected.

Operations that require user approval can trigger high risk attacks. For instance, by forcing a device to connect to a malicious Wi-Fi AP, the attacker can establish the man-in-the-middle attack, redirecting the device to download a malicious app. Or, the attacker can even take complete control of the smartphone by forcing the device to pair with a Bluetooth mouse, which can be used as a remote control. Thus, evading the user approval process is a key success factor of the attacks. One way to evade the use approval process is to display a dialog box with a misleading message, e.g., requesting a user to reconnect to the network, which seems to be caused by the network connection problem. Actual examples of composing such a misleading message will be described in Section IV. Another way is to employ the new attack we developed, *Ghost Touch Generator*, which will be described in Section V. In the next subsection, we will describe the end-to-end attack scenario and its consequences.

C. End-to-End Attack Scenario: A Case for the Malicious Table

We present the end-to-end attack scenario of *Tap 'n Ghost*, which leverages the two attack techniques: *TAP* and *Ghost Touch Generator*. In this scenario, we assume that the attacker implements *Tap 'n Ghost* in a table, which we call *Malicious Table*.

Here are the steps to compromise the victims' smartphone, using the malicious table.

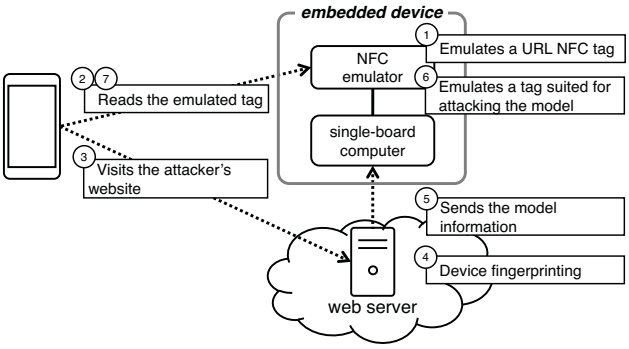

Fig. 3. Overview of the *TAP* attack with the device fingerprinting.

Step 1 An attacker installs a *Malicious Table* in a public space such as a library or a cafe.

Step 2 A victim uses his/her smartphone at the table. The presence of the smartphone is detected by the *TAP* device embedded into the table top.

Step 3 The *TAP* device performs the attack and collects a device fingerprint of the victim's smartphone. Then, it triggers a high-risk action by letting the smartphone read the tailored NFC tags.

Step 4 Upon receiving a pop-up dialog box, the victim will try to cancel the action by tapping the cancel button. The *Malicious Table* will start the *Ghost Touch Generator* attack to alter the selection of the buttons. In case the *Ghost Touch Generator* attack fails, the *Malicious Table* starts over from Step 3 after a certain time interval.

After succeeding the compilation of attacks described above, an attacker can employ the further attacks. For instance, the victim's smartphone will be forced to pair with the Bluetooth mouse emulated by the *Malicious Table*. The attacker can fully take control of the smartphone; for instance, the attacker can install any apps remotely, using a paired Bluetooth mouse. The demo of such an attack is presented in our demo videos, which are available at https://goo.gl/xoVt23.

IV. Tag-based Adaptive Ploy

In this section, we first provide an overview of the *TAP* attack. We then present several techniques to mislead a victim.

A. Overview of the TAP attack

Figure 3 illustrates an overview of the *TAP* attack, which makes use of device fingerprinting. It comprises the two primary components, an NFC tag emulator and a single-board computer with a Wi-Fi controller installed. *TAP* works with a web server, which can be set anywhere connected to the Internet, e.g., a cloud server. We note that the attacker also needs to install a power source. By using the NFC-tag emulator, we can dynamically switch the NFC tags according to the attack scenario.

We now describe how *TAP* works using the example shown in Figure 3[2].

Step 1 First, the NFC tag emulator acts as an NFC tag with a URL data recorded and waits for a victim to approach.

Step 2 When the victim's smartphone comes close to the emulator, it reads the tag and launches a browser to open the URL.

Step 3 The browser then connects to the website specified by the recorded URL.

Step 4 The website employs device fingerprinting by using JavaScript to collect information about the victim's device.

Step 5 The website sends the device fingerprinting information to the onboard computer of TAP. We assume that the computer has Internet access.

Step 6 Upon receiving the device information, the computer determines the tag suited for the victim's device and rewrites the NDEF record of the NFC tag emulator.

Step 7 Finally, the victim's smartphone reads the new NDEF record from the tag and gets attacked again. Note that the smartphone will read a new record after the emulator is turned off (which implies that the old tag went away) and turned on again.

B. Camouflage techniques

In the following, we present *camouflage* techniques, which are established by fully exploiting the NFC stack. These camouflage techniques aim to mislead victims so that they will naturally approve the requests that are triggered by the attacks. They also aim to keep the victims unaware of the presence of the attacking devices. To further empower these camouflage techniques, an attacker can leverage the device fingerprinting. To make the following explanations easy to follow, we first describe a case in which the attacker does not use the device fingerprinting. We will then describe a case in which the attacker needs device fingerprinting.

We present the scenario of a Wi-Fi attack as an example. In this scenario, the goal of the attacker is to mislead a victim into touching the "CONNECT" button when a modified message pops up after reading the malicious NFC tag with the WiFiConfig record. In the Android OS, the format of the confirmation message invoked by the WiFiConfig NFC record is defined in [11]. Since the maximum length of the strings used for specifying an SSID is set to 32 bytes, and the SSID encoding scheme allows the use of the UTF-8 charset [12], the attacker can tweak the SSID strings to deceive a victim.

We show an attack scenario using this trick. The attacker creates a malicious NFC tag with the SSID of WiFiConfig record set to "again." When the victim's smartphone approaches to the malicious tag emulated by the *TAP* system, the following confirmation message pops on the screen:

Connect to network again?

[2]For reference, a code snippet that implements the device fingerprinting is shown in Figure 24 (Apendix). Our demo movies (available at https://goo.gl/xoVt23) demonstrate how the entire device fingerprinting process works in practice.

When the victim notices this message popping up, she or he may think that the Internet connection is lost, that the smartphone is asking to reconnect to the previously connected network, and will touch the "CONNECT" button. Thus, the man-in-the-middle attack is established. Note that a single-board computer can work as a malicious Wi-FI AP. Along with this line, the attacker can create various misleading messages such as,

Connect to network to prevent the data lost?

Such a message will threaten the victim into touching the "CONNECT" button, which again will connect the smartphone to the malicious Wi-Fi AP.

We now turn our attention to the case where the attacker needs device fingerprinting. Through the analysis of 24 Android smartphones equipped with NFC, we found that several vendor customizations use different formats for the confirmation messages[3]. To cope with such differences, the attacker can use the information obtained from device fingerprinting, which was presented in Section IV-A.

In this camouflage attack, an attacker leverages the apps installed in the victim's smartphone; to this end, the attacker specifies "Android Application" in the NDEF record of a malicious tag before presenting the next NFC tag that will pop up a message. After reading the application tag, Android OS will automatically execute the application specified in the record without requiring user approval. The attack aims to make a misleading message that looks real by creating a context. It will lead the victim to think that the message was sent from the installed app, rather than from an invisible attack device.

The attacker first sets an Application NFC tag that launches a popular SNS app, such as Facebook. Subsequently, the Facebook app appears on the screen of the victim's smartphone. The attacker then sets the WiFiConfig NFC tag using the technique described previously. The message popping on the screen appears as follows:

Connect to network ? Facebook app is requesting.

Since the dialog box of this message appears on top of the Facebook app, it looks as if the message is originating from the Facebook app[4]. In addition, Some Facebook users may touch the "CONNECT" button, never knowing that the message is for connecting to a malicious Wi-Fi AP. We provide screenshots of the attacks described above in the Appendix.

V. Ghost Touch Generator

While some users may be trapped with the camouflage techniques described in the previous section, careful users may notice the suspicious behavior and attempt to stop it. In this section, we will introduce a new attack named *Ghost Touch Generator*, which aims to succeed in attacking even when a victim user is careful enough to not get trapped with the camouflage techniques. We will also discuss the attack mechanism insights.

A. Overview

Ghost Touch Generator is an attack that aims to scatter touch events around the original touch area; i.e., even though a victim touches a "CANCEL" button, which should cancel the request to connect to a malicious Wi-Fi AP, the attack makes the operating system recognize the event as a touch of another button, "CONNECT," in a probabilistic way. Thus, the attack can trick the user, with a certain success rate. In the following section, we aim to present the basic mechanism of *Ghost Touch Generator* and reveal the conditions that are needed to establish the attack.

The key idea of *Ghost Touch Generator* is to cause the malfunction by injecting intentional noise signals from the external. Through the empirical observations, we found that we can intentionally cause the malfunction by generating an electric field near the capacitive touchscreen controller, using an electric circuit that can produce large alternating voltage. As we will discuss in Section V-C, applying strong signals causes changes in the current flowing into RX electrodes of the touchscreen controller through the capacitive coupling, and the changes will be detected as the (false) touch events, which are reported at positions where no touch is present.

B. Characteristics of the Attack

To study the conditions that can cause the "false touches," we conduct several experiments using the touchscreen controller that provides raw data collected from the capacitive sensors. In the following, we first describe our experimental setup. Second, we attempt to specify the intrinsic frequency of injected noise signal to maximize the false touches. We then analyze the spatial patterns of the false touch events on the screen with a noise injected at a specific frequency. Finally, we study how an actual touch event by a user affects the spatial patterns of the false touch events. This final experiment will reveal the mechanism of *Ghost Touch Generator*.

1) Experimental setup: Figure 4 shows our experimental setup. Our objective is to measure the effect of noise signals on the behavior of touchscreens. For this experiment, we use the Raspberry Pi 7-inch Touchscreen Display. As an intentional noise signal, we use the sine-wave signal generated by a function generator. We set a copper sheet parallel to the touchscreen controller. This copper sheet is used to create a capacitive coupling with the capacitive sensors. The distance between the sheet and controller was set to 7 cm. We note that the attack can be applied from the rear side of a touchscreen controller, i.e., the rear side of a smartphone.

2) Effect of the frequencies and voltage values: We generate sine-wave noise signals with different frequencies and voltage values. We record raw capacitance values and touch events using the software we developed. Since the touchscreen has 264 capacitance sensors, which consists of a 12×22

[3]For reference, we summarize the results in Table V, Table VI, and Table VII (Appendix).

[4]Note that to create this message, we set the following text string as the SSID: "\u202E.gnitseuqer si ppa koobecaF", where '\u202E' is a Unicode character known as RIGHT-TO-LEFT OVERRIDE.

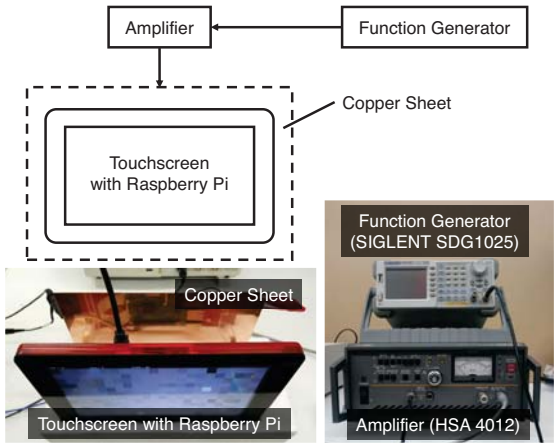

Fig. 4. Experimental setup.

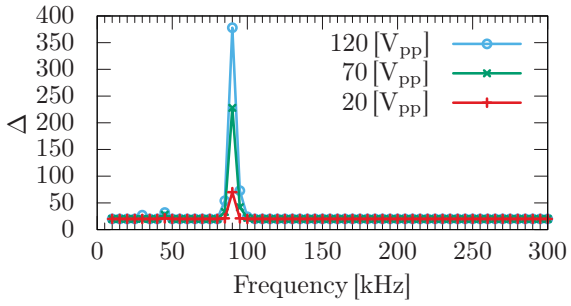

Fig. 5. The effect of different frequencies on touchscreen.

matrix, we can obtain 264-dimensional time-series data. This setup enables us to analyze the spatial patterns of the generated touch events.

To measure the interference intensity on the touchscreen, we introduce a metric, Δ, which is defined as $\Delta = \max_i(\delta_i) - \min_i(\delta_i)$, where $\delta_i = x_i - \bar{x_i}$ and x_i $(i \in \{1, \ldots, 264\})$ is a measured value for each sensor and $\bar{x_i}$ $(i \in \{1, \ldots, 264\})$ is a measured value for each sensor when noise is not injected, respectively. We note that x_i is variable of time; our capacitance logger sampled the raw values at the rate of 7 times per second. In contrast, $\bar{x_i}$ was set as a static value, which was collected when no signal was injected. If no noise signal is applied, Δ becomes roughly 20 when there are no touch events on the screen and Δ becomes greater than 250 when a finger touches the screen. Thus, the metric Δ can measure the impact of noise interference.

We measured Δ, applying noise signal to the copper sheet with three different voltages (20 Vpp, 70Vpp, and 120Vpp) and frequencies, ranging from 5 kHz to 300 kHz. Figure 5 shows the results. We first notice that there are clear peaks at the frequency of 90 kHz. This result indicates that there is a characteristic frequency of noise that can affect the touch controller. As we will study in the next section, this frequency differs for different models of touchscreen controllers. So, specifying the model of the target is crucial to succeeding in attacking. As we have seen, the device fingerprinting technique can be used for this purpose. We also notice that the effect of noise becomes larger with higher voltage in the signals.

3) Spatial distribution of the false touch events: We now study the positions of the touch events caused by the noise signals. In this experiment, nothing touches the screen. Using our monitoring software, we record touch positions for 30 seconds with the sampling rate of two samples per second. The touchscreen has an 800×480 resolution and supports a 10-point multi-touch.

We used three different voltages (20 Vpp, 70 Vpp, and 120 Vpp) and the following two representative frequencies: 60 kHz as a frequency not affecting Δ and 90 kHz as a frequency affecting Δ the most. As expected, the touchscreen does not report any touch events with the 60 kHz frequency. In the followings, we omit the results of 60 kHz frequency. Figure 6 shows the results for 90 kHz frequency. First, we notice that the touchscreen controller did not recognize touch events when the voltage was set to 20 Vpp. We also see that higher voltage signals cause false touch events more frequently. Second, we see intrinsic spatial patterns of touch events, i.e., they linearly spread out on the screen[5]. We also see that many touch events are focused on the top or bottom edges of the screen panel. These observations indicate that even if an attacker waits for a long time, it seems unlikely that a false touch is fired at target coordinates with a high probability, given the skewed spatial distribution.

4) Limiting the dispersion with a real touch event: After several trials, we found that touching on a screen can fix the skewed spatial distribution of false touches. Although not conclusive due to the "black box" nature of the touchscreen controllers, we conjecture that the touching with a finger stabilizes the area of capacitive coupling. The good feature of this phenomenon is that while touching on a screen makes the distribution focused on a certain area, it still keeps scattering the touch events; thus, it can create false touch events in a more predictable way.

We repeated the similar experiments but added a finger touch this time. Figure 7 shows the experiment results. Under the low voltage signal of 20 Vpp, the false touch events occur only if a finger touches the screen. More importantly, we can see that the positions of the false touches are centered on the line where the true touch point is located. It was also seen that the direction of the line did not change even when we placed the touchscreen at different angles, suggesting that the attack is tolerant to the random placement of the device. The line was formed along the RX electrodes of the touchscreen consistently; the mechanism of the phenomenon will be discussed later. These are desirable characteristics because usually, GUI buttons are aligned in a row; e.g., CONNECT/CANCEL, YES/NO, or OK/CANCEL. Therefore, an attacker can expect that a touch event will be scattered on

[5]As we will see in the next section, the direction of the spread patterns differs for different models of touchscreen controllers; i.e., horizontal spread or vertical spread.

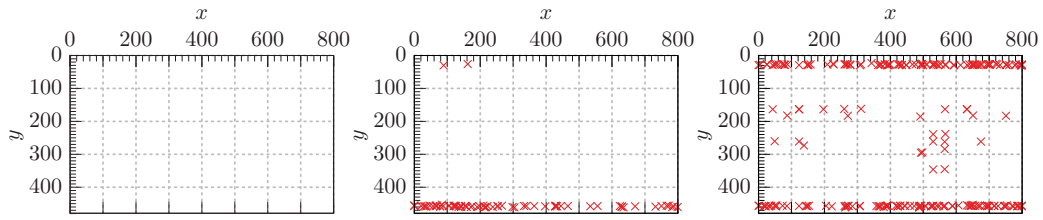

Fig. 6. Coordinates of the touch points reported by the touchscreen controller. The injected signals had three different voltage values. The frequency was set to 90 kHz. **Left:** 20 Vpp, **Center:** 70 Vpp, **Right:** 120 Vpp

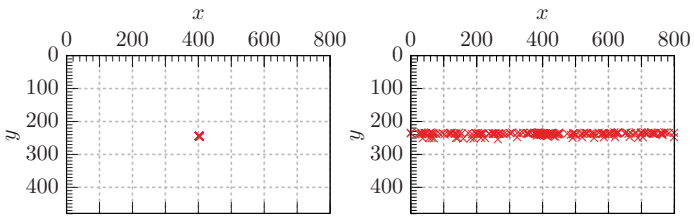

Fig. 7. Coordinates of the touch points reported by the touchscreen controller. While the experiment a finger keeps touching the point centered on the screen. **Left:** no signal is applied. **Right:** a signal with 20 Vpp and 90 kHz is applied.

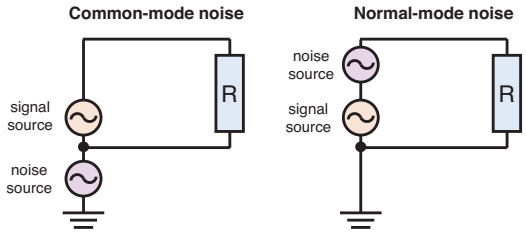

Fig. 8. Common-mode noise and normal-mode noise.

a wrong button, with a probability of $1/2$, with an assumption that the touch events are uniformly scattered along a line. We note that screen orientation also matters. If a screen is in portrait mode, scattered touch events along the vertical line may not produce a touch on the targeted button. As we show in Section VI, the direction of scattered touch events differ among the different models. By making use of the device fingerprinting techniques, an attacker can obtain the information about the model as well as the current screen orientation; these information will be used to check whether or not *Ghost Touch Generator* is effective.

C. Attack Mechanism Insight

As shown in Figure 8, based on how a noise source is connected to a circuit, electric noise can be classified into the two types: common-mode noise and normal-mode noise. While common-mode noise does not affect the voltage across a passive element such as a resistor and a capacitor, normal-mode noise does change it. It is known that a touchscreen of a smartphone can malfunction due to the normal-mode noise signals, which are leaked from the smartphone's battery charger or LCD screen [13]. The phenomenon is caused by the fact that normal-mode component of the noise signals will affect δQ, which is proportional to the amount of current changes measured with the touchscreen controller; i.e., $\delta Q = C \times \delta V$. Recall that C denotes the capacitance between the TX and RX electrodes and δQ denotes the amount of current changes. δV denotes the changes of the voltage across the capacitor; i.e., δV is the voltage induced by the normal-mode noise. Thus, even though the capacitance, C, has not been changed, the touchscreen controller will detect the change of current as a false touch.

Touchscreen controller manufacturers have developed countermeasures against the electromagnetic interference (EMI) caused by the weak noise, which is applied to a touchscreen controller unintentionally from the internal circuit; e.g., those emanating from a battery charger or an LCD screen. However, as we have experimented, if a strong noise signal is intentionally applied to a touchscreen controller, it will affect δQ, and the change will be detected as a false touch. The key idea of *Ghost Touch Generator* was to cause an intentional malfunction by injecting intentional noise signals externally. We found that producing large alternating voltages at a specific frequency near a touchscreen can cause a malfunction through capacitive coupling with the RX electrodes.

Based on the discussion so far, we now attempt to answer the following research questions, which are raised through our experiments.

- *How does "a real touch event" change the dispersion area, i.e., from Figure 6 to Figure 7?*
- *Why are the invoked false touches scattered along the RX electrodes, which are actually touched?*

Figure 9 shows the two diagrams of the circuits that express the high-level behavior of a touchscreen controller when the external noise signal is injected. Unlike Figure 1, there is a noise source between the drive voltage source and the ground. This noise source expresses the alternating current applied to the copper sheet, which is electrostatically coupled with the touchscreen. When an RX electrode is not touched by a finger (left circuit of Figure 9), the noise source will not affect the voltage measured at the RX electrode because the noise source only changes the ground voltage of the entire touchscreen circuit. The circuit is similar to the one shown in the left circuit of Figure 8, i.e., the applied noise appears as common

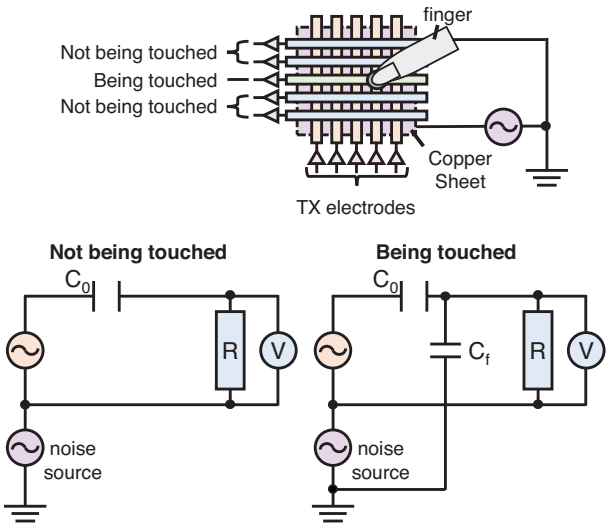

Fig. 9. Diagrams of the circuits that express the high-level behavior of a touchscreen controller when noise signal is injected: **Top:** the circuit of a touchscreen controller, **Left:** the equivalent circuit when an RX electrode is not touched by a finger, **Right:** the equivalent circuit when an RX electrode is touched by a finger.

mode. Therefore, the δQ will not change, and false touches will not appear on the corresponding RX electrodes.

In contrast, when an RX electrode is touched by a finger, the noise source will affect the voltage measured at the RX electrode, which is touched by the finger. As shown in the right circuit of Figure 9, the noise source is connected to the RX electrode through the capacitance of the human body. The circuit is similar to the one shown in the right circuit of Figure 8; i.e., normal mode noise is applied to the circuit. Therefore, the δQ will change, and false touches will appear on the corresponding RX electrodes.

The above observations lead us to the following conclusion; the external noise signals injected by the *Ghost Touch Generator* affect the current measured at each RX electrode in a different way; i.e., appears as common mode noise when an RX electrode is not touched by a finger or normal mode noise when an RX electrode is touched by a finger. Thus, false touches appear only on the lines corresponding to the RX electrodes which are actually touched. This conclusion is supported by the fact that the false touch line in Figure 7 is in parallel with the RX electrodes of the actual touchscreen we used in the experiment.

VI. Technical Feasibility Assessments

To evaluate the technical feasibility of *Tap 'n Ghost*, we performed two studies, which aim to verify that NFC tags embedded inside a common object can be actually read by smartphones and to verify the success of *Ghost Touch Generator* attack.

A. Maximum NFC Reading Distance

For this study, we use 24 Android smartphones/tablets, which are manufactured by the 12 different vendors. We study the maximum NFC reading distance of the smartphones to demonstrate the validity of the idea of embedding malicious NFC tags in a physical object. An NFC tag is attached to the backside of the wood board of the walnut material. We read the tag using the smartphones placed on the backside. We measured the maximum communicable distance by changing the thickness of the wood board at intervals of 5 mm and recording the success of reading the tag. We found that the maximum NFC reading distance was 3.4 cm in average. The maximum and minimum of the measured distance were 5.0 cm and 2.0 cm, respectively[6]. If we consider the thickness of a table top, we can conclude that the measured maximum distance is large enough to establish the attacks by *Tap 'n Ghost*.

B. Conditions of the successful Ghost Touch Generator.

We now empirically study the conditions for the successful attacks. For this study, we use 7 Android smartphones/tablets listed in Table II[7]. As our amplifier is not capable of generating voltage greater than 150 Vpp, which was smaller than the effective voltage needed to successful attacks, we used a high-voltage transformer taken out of a plasma ball. When the attack device generates the highest/lowest voltage (700V/30V) shown in Table II, it consumed 26W/6W, indicating that the power consumption of the attack device is small.

Figure 10 shows the setup of the experiments. The smartphone and the copper sheet are insulated with the polycarbonate plate of 5 mm thick. We first employ device fingerprinting by following the procedure shown in Section IV. The fingerprint information can be used to identify the characteristic frequencies for the smartphones to cause the malfunctions. For the smartphones that had caused malfunctions, we will further test the following tasks.

We created an NFC tag that requests the Bluetooth pairing. A smartphone that read the tag will pop up a dialog message, "Are you sure want to pair the Bluetooth device? (NO, YES)." We then touch the button of "NO." Before the smartphone reads the tag, we have applied *Ghost Touch Generator*. We will see whether the actual touch becomes "YES" (attack succeeded) or "NO" (attack failed).

Sometimes, we do not see any responses even though we touch the button due to the noise injection. In such cases, if there are no responses back after the five consecutive touches, we count it as a failure of the attack. Also, if the patterns of the touch scattering for a device has a horizontal/vertical direction, we set the orientation of the device to portrait/landscape.

Table II summarizes the results. For the 5 out of 7 models, we specified the characteristic frequencies and voltage values

[6]For reference, the full results are shown in Table VII (Appendix).

[7]We rented 17 devices for the experiment of measuring NFC reading distance, however, we were not able to use these devices for the experiments of *Ghost Touch Generator* because applying the attack has a risk of causing physical damages on the devices.

TABLE II
THE CONDITIONS OF THE SUCCESSFUL *Ghost Touch Generator* ATTACK. THE DIRECTION OF THE SCATTERING PATTERNS IS DEFINED WHEN A SCREEN IS SET IN PORTRAIT MODE.

Device	Manufacture	Success false touches	Frequency [kHz]	Voltage [Vpp]	Success attack rates	Scattering patterns
Nexus 7	ASUS	✓	128.2	40.0	18/30	vertical
ARROWS NX F-05F	FUJITSU		—	—	—	—
Nexus 9	HTC	✓	280.9	490.0	0/10	horizontal
Galaxy S6 edge	SAMSUNG		—	—	—	—
Galaxy S4	SAMSUNG	✓	384.5	70.4	13/30	horizontal
AQUOS ZETA SH-04F	SHARP	✓	202.0	700.0	0/10	horizontal
Xperia Z4	SONY	✓	218.0	340.0	20/30	horizontal

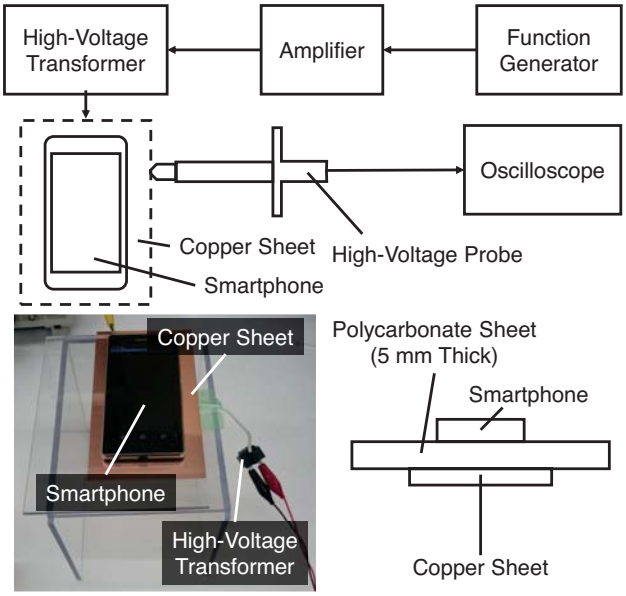

Fig. 10. Block diagram and photos of setup for observing effect of alternating current on off-the-shelf smartphones

that can cause malfunction, i.e., "false touch." Of the 5 models that cause malfunctions, 3 models succeeded in attacking with probabilities distributed around $1/2$; i.e., the OS detected the touch for a wrong button and the device was paired with a Bluetooth device. The rest of 2 models worked as follows. For Nexus 9, the detected touch events were biased to a specific area, which was not close to the buttons; thus, the attacks failed. For AQUOS ZETA SH-04F, when a finger touched somewhere in the right/left half of the screen, the false touches appeared on the left/right half on the screen; thus, the attacks failed. Thus, the patterns of false touches depend on the models. There were two models that did not generate false touch events; the one that the detected touch events lag behind the finger's touch (Galaxy S 6 edge) and the one that does not recognize the touch at all (ARROWS NX F-05F). We also found that Ghost Touch Generator works at a distance much longer than maximum NFC reading distance – 5.0 cm. This observation indicates that the Ghost Touch Generator will succeed within the range in which the TAP attack succeeds[8].

TABLE III
DEMOGRAPHY OF RESPONDENTS (ONLINE SURVEY).

	Age (Years)		
# gender	10–29	30–49	50–
F: 149 M:150 Other: 1	107	169	24

VII. PRACTICALITY OF THE THREAT

In this section, we first evaluate the practicality of the threat using a *Tap 'n Ghost* attack by testing the three assumptions made concerning a victim of this attack:

- **A1**: Has an Android smartphone equipped with NFC,
- **A2**: Has enabled NFC functionality on the smartphone, and
- **A3**: Has unlocked the screen of the smartphone when she or he brings the smartphone close to a *Tap 'n Ghost*-installed object.

We then assess the real world impact of the attack based on the statistics derived from our user study and discuss the cost needed to perform the end-to-end attack. Finally, we describe two possible attack scenarios to clarify the practicality of the attack.

A. Validity of **A1** (Device constraint)

The assumption **A1** limits the target of the attack to Android smartphones/tablets. We note that as the share of Android in the smartphone OS market is 85% [14], the number of potential target devices is large.

Online survey: To test the assumption **A1**, we performed an online survey with 300 participants. We recruited participants who own Android smartphones. In Table III, the population of the participants is shown. We used a crowdsourcing platform to recruit them. To complete the full survey, it took roughly two days.

In the survey, we asked whether or not the participants' smartphones were equipped with NFC. To ensure the accuracy of the response, the questionnaire included a description that showed how to confirm the presence of NFC functionality on a smartphone. More specifically, we put a question that asked participants to report their device names/versions, which we used to check the validity of responses. Of the 300

[8]Interested readers can refer to the movies (available at https://goo.gl/xoVt23) for clarification.

participants, 214 participants answered that their smartphones were equipped with NFC. We also analyzed the product names of smartphones reported by the participants, and found that their responses were consistent.

In addition to the observations derived from the online survey, there are several facts that support the assumption **A1**. It is forecasted that NFC market will grow at the rate of 17.5% during 2018 to 2023 due to the increased volume of mobile/contactless payments [15]. Also, it has been forecasted that the shipments of Android NFC-enabled smartphones will reach 844 million in 2018 [7]. These facts indicate that the potential target of *Tap 'n Ghost* attacks is increasingly becoming ubiquitous.

B. Validity of **A2** *(NFC availability)*

Tap 'n Ghost will not succeed unless the NFC is enabled on the victim's smartphone. To verify the assumption **A2**, we manually investigated 24 smartphones listed in Table VII. We found that the NFC was enabled in the factory settings in 16 out of 24 models. Interestingly, in more recent models, the NFC is enabled in the factory settings. We also note that the NFC-based mobile payment has become increasingly popular over the past several years. For instance, a report [16] predicts that in 2021, NFC or other mobile payment technologies will generate close to 190 billion U.S. dollars in transaction value. Another report [17] reports that in 2018, more than one-third of smartphone users ages 14 and older will use a smartphone to pay for a purchase at a POS at least once every six months. The widespread of NFC-based mobile payment will incentivise smartphone users to always turn on the NFC functionality on their devices.

Online survey: To complement the observations on the NFC availability, we tested our assumption by using an online survey, which was a continuation of the previous one. To the participants who answered that their smartphones were equipped with NFC in the previous question, we asked them when they usually enable the NFC functionality. Of the 214 participants, 48 participants answered that they always enable the NFC functionality, and 11 answered that they occasionally turn on the NFC functionality. In total, roughly one-fourth of the participants with NFC-powered smartphones made use of NFC functionality in their daily lives.

C. Validity of **A3** *(Human behavior)*

To verify the assumption **A3**, which is a factor related to human behavior, we conducted a user study, following the threat model shown in Section III.

Scenario: In this experiment, we simulated a situation where a person is studying or doing paperwork at a public space such as a library or a cafe, and an attacker has embedded a *Tap 'n Ghost* system in a table installed in the library or cafe. While the person is studying, she or he may want to use a smartphone for several purposes, for instance, looking up a word, searching on the web, or using a calculator app. We also simulated a situation where a person with a smartphone is taking a break at a table.

TABLE IV
DEMOGRAPHY OF THE PARTICIPANTS (USER STUDY).

	Generation		
# Gender	10's	20's	30's
F: 7 M: 9	5	10	1

Design/policy of the study: Our experiment is a deceptive study by nature, which means that we did not inform participants in advance that our true objective was to test the assumption **A3**. Thus, they did not know that there were some equipment embedded in the table on which they were performing the given tasks. We designed our experiments such that the participants were treated fairly and with due consideration of their rights. In light of this policy, we did not install any malicious NFC tags nor *Ghost Touch Generator*. Instead, we tried to test the assumption in a non-intrusive way; that is, we embedded multiple NFC reader/writers into a table to sense the status of NFC-equipped smartphones. Using the measurement technique described later, we can determine whether the smartphone is locked or not when it is brought close to the measurement system. We note that as shown in Section VI, *Ghost Touch Generator* will succeed within the range in which the TAP attack succeeds; i.e., if a unlocked smartphone is brought close enough to be sensed with our measurement system, the smartphone will be successfully attacked by *Ghost Touch Generator* with the succeeding probability shown in Table II.

Participants: We recruited 16 participants who owned NFC-equipped Android smartphones. To search for participants, we used a web-based announcement system of our institution. An interested person directly contacted us using e-mail as a communication channel. Before enrolling a participant into the experiment, we checked whether they own NFC-equipped Android devices in the following way. We first asked participants whether their phone has NFC functionality and if yes, what their settings were. To do so, we provide an instruction so that they can check it easily.

Table IV shows the population of the participants. They were undergraduate or graduate students studying at a university. Of the 16 participants, only one majored in computer science; the remaining students majored in various fields such as other science and engineering disciplines, literature, sports science, education, and commerce. The reward for the participants was 20 USD per person. The length of entire session was roughly 40 minutes per participant.

Experimental Setup: We have developed the smartphone status monitoring system, which consists of 16 NFC reader/writers and a laptop connected to them (see Figure 11). This system is embedded in a wooden table. For reference, detailed technical descriptions of the system are shown in Appendix. This monitoring system can be used to test the assumption **A3**; hence, it senses an NFC-equipped smartphone put on the table and detects whether the smartphone is unlocked (attackable) or locked (un-attackable) by analyzing the NFC communication. As the system uses NFC communication as

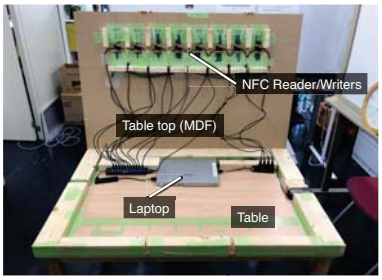

Fig. 11. Smartphone status monitoring system embedded in a wooden table.

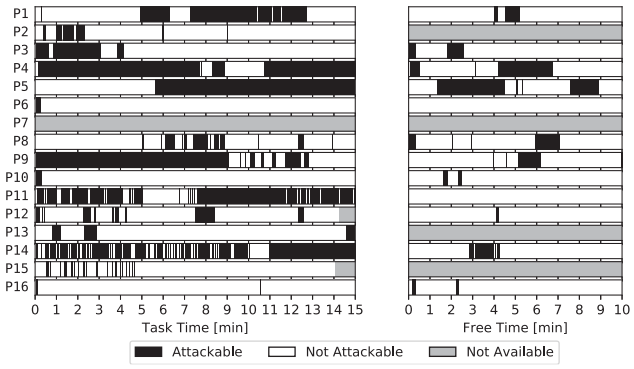

Fig. 12. Results of user study.

a means of detecting the status of a smartphone, it is clear that a smartphone detected as unlocked can be attacked with the *Tap 'n Ghost*. We note that this monitoring system can be turned into a *TAP*-installed table if we configure it as such.

User study procedure: The experiment consisted of the three sessions:

Session 1 Quiz (15 mins)

In this session, the aim was to simulate a situation in which a person had to use a smartphone while she or he was working on a task at a public place. Participants were asked to write the answers to the quizzes written on a paper. At the time of recruiting, we informed the participants that the objective of this experiment was to observe how a person makes use of a smartphone when she or he needs to search for information. As we focused on testing the assumption **A3** in this study, we designed our experiment assuming that other assumptions **A1** and **A2** had been established; these two assumptions have been verified in the previous subsections. To achieve this condition, we asked the participants to enable NFC functionality during the experiment. To make the context of NFC natural, we informed the participants that NFC is used for recording check-in/out times and asked them to touch the NFC tags we provided with their smartphones at the beginning and ending of the experiment. We note that we did *not* inform users to unlock the smartphone (except when they needed to unlock the smartphone to read the check-in/out NFC tags), because our objective was to check whether or not a person brings their device close to the table in an unlocked status.

During the session, participants used their smartphones when they answered the quiz questions. To emulate a scenario described before, we created quizzes such as "Write the names of the three kings who owned their pyramid complex at Giza." Consequently, most of participants had to search the Internet to get the correct answer; it will emulate the situation in which a person is studying at a public place and used their smartphone to lookup something.

Session 2 Break time (10 mins)

In this session, the aim was to simulate a situation in which a person who brings a smartphone is seated at a table. In contrast to the previous session, we did not force the participants to use their smartphones. After the Session 1 concluded, we asked participants to remain seated at the table while we were preparing for the next session and left them the room. We provided them with tea and snacks to put them in a relaxed state.

Session 3 Debriefing (10 mins)

In the debriefing session, we first disclosed the true purpose of this study and the reason why we had to deceive the participants. We did not receive any negative feedback on the deceptive aspect of the study. We then asked a series of questions to understand how they actually behaved during the experiment. We also asked them to examine their smartphones to get detailed spec information.

Results: In Figure 12, the results of these activities are shown. We first note that there were several exceptions (marked as "not available") as follows. Among the 16 participants, P7 owned a device that did not respond to the probes sent from the NFC readers. The device was a Huawei Honor 8. When it receives a probing command, it pops up a window of Huawei Pay without sending back any commands. As we will discuss in Section VIII, this type of user approval process can be an effective countermeasure against the threats caused by a malicious NFC tag. P2, P13, and P15 misunderstood our instruction and disabled NFC after they completed the first session. We also note that P12 and P15 completed the quiz task within 15 mins. In the following, we eliminate the corresponding "Not Available" time marked with grey color in Fig. 12 from the analysis.

In the Session 1, all the participants presented opportunities to be attacked within 15 mins. Most of the participants reported in the debriefing session that she or he had unlocked her or his smartphone and put it on the table when she or he looked up something using the smartphone. As the participants needed to work on quizzes, it is natural that they behaved in that way.

In the Session 2, 10 out of 12 participants presented opportunities to be attacked within 10 mins. Interestingly, while the participants were taking a break, majority of them used their smartphones on the table. This result indicates that *Tap 'n Ghost* is also effective for person who is not studying, but taking a break at a table. Two participants who did not present opportunities to be attacked during the break time reported that they were either eating snacks or using their smartphones

keeping a distance from the table top. We note that there were no participants who noticed that there was our measurement system inside the table.

Summary: To summarize, the results obtained through our user study demonstrates that there are scenarios in which the assumption **A3** holds; i.e., in our experiments, most of participants presented opportunities to be attacked during the sessions.

D. Attack Impact

Based on the survey results, we assess the real world impact of *Tap 'n Ghost*.

Number of potential target devices: First, we attempt to estimate the number of devices/users that could be compromised by our attack. The number of NFC-enabled Android smartphones will reach N=844 million by 2018 [7]. We use this number as a baseline. As we have shown in the validity of **A2**, we revealed that roughly $q_1 = 1/4$ of users with NFC-enabled Android smartphones turn on NFC daily. Also, in the validity of **A3**, we revealed that $q_2 = 10/12$ of participants exhibited attackable opportunities when they use smartphones at a table in which Tapn Ghost could be installed (free time scenario). Assuming that $q_3 = 1/10$ of users may use their smartphones at a public table, the rough estimation of the attackable target devices is $N \times q_1 \times q_2 \times q_3 = 23.4$ million, which is a significant number. We note that the estimate of q_3 may be conservative.

Success probability of a single attack: Next, we attempt to estimate the success probability of a single attack for the attack scenario described in Section III-C (*Malicious Table*). To succeed in attacking, the following conditions must be satisfied:

- C_1: The smartphone comes with Android OS.
- C_2: The smartphone is equipped with NFC.
- C_3: The victim has enabled the NFC functionality.
- C_4: The smartphone's touchscreen controller is attackable with *Ghost Touch Generator*.
- C_5: The victim has unlocked the smartphone when she or he brings it close to the *Malicious Table*.
- C_6: *Ghost Touch Generator* attack has succeeded.

In the followings, for each of the above conditions, C_i, we attempt to estimate the probability, p_i that the condition is satisfied.

As we have shown in the validity of **A1**, Android's share in the smartphone OS market is 85% [14]. Therefore, we can estimate p_1 as $p_1 = 0.85$. Furthermore, as we already have shown in the validity of **A1** and **A2**, we can estimate p_2 and p_3 as $p_2 = 214/300$ and $p_3 = 1/4$, respectively. From the results shown in Table II, we estimate p_4 as $p_4 = 3/7$. As we have shown in the validity of **A3**, p_5 can be estimated as $p_5 = 10/12$ (free time scenario). Finally, as discussed in Section V-B, **C6** will be satisfied with a probability of $p_6 = 1/2$. The success probability of a single attack is calculated as the joint probability; i.e., $p = p_1 \times p_2 \times p_3 \times p_4 \times p_5 \times p_6 = 0.03$.

Overall attack success probability: Although the success probability of a single attack is not high, there are several opportunities to carry out an attack on each victim as we have shown in Figure 12. That is, an attacker can retry the attack. In addition, as a *Malicious Table* is installed in a public space, the attacker can target many users. That is, even if the attacks on a victim were all failed, the attacker could wait for another victim who will take a seat at the table.

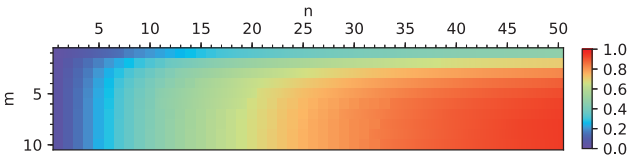

Fig. 13. The attack parameters (n, m) vs. attack success probability.

Based on the observations, we estimate the probability that the *Tap 'n Ghost* attack is succeeded at least once within a period of time. Here is a generalized scenario:

- n persons with their smartphones will take a seat at the *Malicious Table* within a period of time (say, one day).
- For each person, the *Malicious Table* can perform the attack m times.

The overall attack success probability is estimated as

$$p_A(n, m) = \left(1 - (1 - \prod_{j=1}^{5} p_i)^n\right) \left(1 - (1 - p_6)^m\right),$$

which is a probability that the attack is succeeded at least once within a period of time.

Figure 13 shows how the parameters (n, m) affects the attack success probability. As we see, increasing the number of retries, m, effectively increases the attack success probability. For instance, if an attacker can retry the Ghost Touch Generator for $m = 3$ times, she or he can establish the success probability of 0.71 when $n = 30$ persons visit the table within a period of time. We also note that the attacker can increase the number of the *Malicious Tables* if they want to accelerate the speed of succeeding attacks. The cost of implementing the attack device will be discussed in the next subsection.

E. Implementation Cost

To implement a *Malicious Table*, an attacker needs to build two systems, *TAP* and *Ghost Touch Generator*, and install them under the table top. A *TAP* device consists of the following devices: NFC reader/writers, a small computer, and a battery pack. For the user study described in Section VII, we used 16 NFC readers/writers and a laptop PC to implement a *TAP* device. For the real attack setup, we can replace the laptop PC with a small computer, such as Raspberry Pi. A *Ghost Touch Generator* consists of the following devices: a DDS signal generator, a high-voltage transformer, a copper sheet, a small computer, and a battery pack. Note that a small computer and a battery pack can be shared with the *TAP* device.

In total, the cost for implementing a *Malicious Table* is roughly 490 USD, i.e., 320 USD for 16 NFC reader/writers, 20 USD for a DDS signal generator, 6 USD for a high-voltage

transformer, 4 USD for a copper sheet, 40 USD for single-board computer, and 100 USD for a battery pack.

F. Attack Scenarios

Finally, we describe two possible attack scenarios of *Tap 'n Ghost*. We also discuss the feasibility of the attack on the basis of the attack success probabilities we derived.

Attacking random targets: In this scenario, an attacker will set up a malicious table at a public space such as a 24/7 dining cafe or library. An attacker can expect to have many potential victims take a seat at a table and use their smartphones on it; which will lead to the success of the attack. Following the attack success rate, shown in Figure 13, an attacker can increase the expected number of attack successes, as she/he leaves the table for a long time. For instance, if we assume $m = 10$, the attack will succeed, with the probability close to 1, for at least one victim out of $n = 30$ mass targets. As the attack setup keeps running, the number of attack successes will linearly increase over time.

Attacking the specific targets: In this scenario, an attacker will set up a malicious desk at a targeted place, such as an office or open space of a company. The attack targets a specific group of people or individuals, rather than randomly-selected individuals who visit a public space. An attacker will bring a pre-installed malicious desk to the office, by pretending to be a freight forwarder. Alternatively, an attacker can ship pre-installed malicious desks, as an online equipment supplier. Of course, the setup requires an additional cost. However, as this is a targeted attack, even the success of a single attack has significant value for an attacker, e.g., stealing the confidential information of a company, or succeeding in an impersonation attack, in order to further employ an advanced and persistent threat attack. Again, the attack success probability can be seen in Fig. 13. This time, the number of potential targets will be limited as the malicious desk is placed in a private space. However, as the group of targets will have several opportunities to take a seat at the table in several days, an attacker can expect to have large n over time; e.g., the targets will take a seat at the table for $n = 30$ times in total, which is large enough to establish the high attack success probability.

VIII. Discussion

In this section, we first discuss the possible defenses against the threats of the *Tap 'n Ghost* attacks. Next, we discuss the ethical considerations.

A. Countermeasures

We now discuss possible countermeasures against the threat of *Tap 'n Ghost* attacks. We divide the discussion into three groups according to three key components.

NFC: The simplest and the most effective defense is to add/improve the user approval processes before Android OS launches applications recorded in a tag. The user approval processes adopted by iOS can increase the security of NFC-based services, i.e., the latest iPhones (iPhone XS, XS Max, and XR) that perform operations written in NFC tags only when user approval is provided[9]. We note, however, by adding the extra user approval process, the usability of NFC-powered services has been sacrificed to some extent. A good example of establishing both usability and security for the approval process of NFC service is Huawei Pay, which requires a fingerprint authentication for each NFC payment transaction.

Even though a usable approval process for reading NFC tags is widely introduced in future, the threats caused by a deceptive pop-up message remains. To mitigate such threats, Android OS should change the format of messages associated with NDEF records. By explicitly presenting the reason why an operation is invoked, it will thwart the threats caused by deceptive messages.

Touchscreen: While conducting the experiments described in Section VI, we noted that some touchscreen controllers stopped working when a strong electric field was applied. Although these observations are not conclusive, we conjecture that the manufactures of these controllers may have installed mechanisms to stop the controllers upon detection of external noises. In fact, as Ref. [13] reported, manufactures of touchscreen controllers have developed techniques for dealing with the noise that can interfere with capacitive touch sensing. Incorporating such mechanisms will lead to eliminating the threats of *Ghost Touch Generator*. In addition, as Kune et al. proposed in Ref. [18], there are several analog/digital countermeasures against intentional EMI attacks, e.g., a filter that attenuates external noise signals and signal processing to eliminate anomalous inputs. These techniques will also be useful as countermeasures against the threats of *Ghost Touch Generator*.

Objects: It is almost impossible to visually detect a *Tap 'n Ghost* system because it is embedded into physical objects. However, there may be situations where law enforcement agencies want to inspect physical objects such as desks inside a building to investigate whether a *Tap 'n Ghost* has been installed. An active probe that searches for NFC tags should be developed to make this task easier. For this purpose, it is also possible to build a *Tap 'n Ghost* honeypot that behaves as an NFC-enabled smartphone. The drawback of this approach is that it is not scalable because the practical working distance range of NFC is at most about 4 cm. Further research is needed to shed more light on this problem.

B. Ethical considerations

As several researchers have reported, the threat of attacks using malicious NFC tags is publicly known [1]–[6]. Several groups such as the NFC Security Awareness Project and W3C have addressed the danger of reading unknown NFC tags [5], [19]. The objective of our work was to explore the threats of malicious NFC tags by embedding them into common objects. As we have demonstrated, the threats of malicious NFC tags become further viable through *Tap 'n Ghost*. On the other hand, our work studied the threat of active attacks against

[9]Older iOS smartphones read NFC tags only when they receive requests from the foreground apps.

touchscreens. We are taking active steps to notify related vendors of the risk with the aid of JPCERT/CC. In addition, we provide possible countermeasures that will remedy the threats. We hope that our paper will be informative to further enhance the security of NFC-powered smartphones and touchscreens.

All our experiments involving human subjects followed the guidelines provided by our institution. Although our experiments were not designed to collect any privacy-sensitive information, we provided participants with an informed consent such that they could make a decision as to whether or not they wanted to be involved. We were concerned that participants of our user study would not behave naturally if we disclosed the true objectives of the experiment in advance. Therefore, we carefully designed a deceptive study so that we can minimize any risks. All the participants were throughly debriefed at the end of the study. We did not receive any negative feedback from the participants.

C. Open research question

As we mentioned in the introduction section, our work is the first to demonstrate the threat of active attack against touchscreens. Along this line, there is an interesting open research question. As the touchscreen has become a standard user interface that is now used for several critical applications, such as e-voting, equipment manipulation at factories, and public transportation systems, the results obtained through our research imply that we need to have mechanisms that can verify the correctness of the analog signal that serves as input to touchscreen-equipped systems. Thus, how externally injected signals affect the generic touchscreen interfaces and their possible countermeasures remain an open research question. There are many promising avenues for further research.

IX. Related Work

Attacks using NFC: There have been several studies on the threats of attacks using NFC technology [1]–[6]. Miller [4] reported that malicious NFC tags can attack browser exploits and NFC stack bugs that existed at the time. Gold et al. [6] demonstrated a phishing attack that uses a smart poster with malicious NFC tag attached. The accessed website prompts users to log in to a fake SNS site. They also demonstrated that an attacker can write a malicious file to the victim's device by using the peer-to-peer mode of NFC. Wall of Sheep [5] demonstrated the experiment using NFC tags attached to smart posters and buttons at the DEFCON venue. At the venue, they put posters that say "Find a Wall of Sheep button and scan it with your NFC phone for exclusive discounts, tools and surprises every day." They reported that about 50 attendees scanned the NFC tags that "could" have been malicious tags. These studies assumed that an attacker can come close enough to a victim, or the victim intentionally read the malicious NFC tag by using posters or other existing facilities.

Our approach is different from these prior studies in that ours can achieve complex and tailored attacks by leveraging NFC card emulation.

RFID tags: NFC is a specialized subset within the family of radio frequency identification (RFID) technology. Several researchers have studied the risk of RFID tags that can be attached to various things [20], [21]. Baldini et al. [20] reported the application of RFID tags in the retail sector and discussed associated privacy issues and countermeasures. Juels published a survey paper on the research of privacy and security of RFID [21]. The survey examined the privacy protection mechanisms and integrity assurance in RFID systems. In the paper, Juels mentioned the importance of *user perception* of security and privacy in RFID systems as users cannot see RF emissions. The indication is closely related the problem we addressed in this paper.

Attacks on touchscreen: There have been many studies on the side-channel attacks on touchscreens (LCDs); Aviv et al. [22] used smudge left on the screen to infer a graphical password, Maggi et al. [23] used the data collected from a surveillance camera to recognize keystrokes of a victim, and Hayashi et al. [24] used electromagnetic emanation to reconstruct a victim's tablet display. To the best of our knowledge, while these attacks passively steal data from the touchscreen, our *Ghost Touch Generator* is the first attack that actively radiates signals toward touchscreen to cause targeted malfunctions.

We note that there have been several works that studied how electromagnetic interference (EMI) affects devices. For instance, Kune et al. [18] studied the susceptibility of analog sensor systems to signal injection attacks by intentional, low-power emission of chosen electromagnetic waveforms. While their study attacked implantable medical devices and consumer electronic devices containing microphones, ours is the first to attack touchscreen controllers with EMI.

X. Conclusion

We introduced a novel proof-of-concept attack named *Tap 'n Ghost*, which targets NFC-enabled smartphones. To fully explore the threats of *Tap 'n Ghost*, we developed two striking techniques: *Tag-based Adaptive Ploy* and *Ghost Touch Generator*, which enable an attacker to carry out various severe and sophisticated attacks without being perceived by the device owner who unintentionally puts the device close to a *Tap 'n Ghost*-installed object. Through the extensive experiments using off-the-shelf smartphones, we demonstrated that the proposed attacks work in practice. Besides, our study including an online survey and a user study demonstrated that the threat caused by the attack is realistic. Although our attack is a proof-of-concept, we provide possible countermeasures that will thwart the threats. We believe that the concept of our attacks sheds new light on the security research of mobile/IoT devices.

Acknowledgment

We thank our shepherd, Mike Reiter and the anonymous reviewers for providing us with valuable comments. We thank Yasushi Matsunaga, Yoshimichi Ohki, and Kazuyuki Ishimoto for sharing the valuable comments on the mechanism of Ghost Touch Generator. We also thank Makoto Gotoh for granting

permission to use the several equipment, and Hiroki Sudo for assisting the NFC-related experiments. We are grateful to N. Asokan for providing us with useful comments and feedback to the earlier versions of this manuscript. Finally, we thank Shigeki Goto for his various supports in carrying out this work.

REFERENCES

[1] M. R. Rieback, B. Crispo, and A. S. Tanenbaum, "Is your cat infected with a computer virus?" in *Pervasive Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on*. IEEE, 2006, pp. 10–pp.

[2] C. Mulliner, "Vulnerability analysis and attacks on nfc-enabled mobile phones," in *ARES*, 2009, pp. 695–700.

[3] R. Verdult and F. Kooman, "Practical attacks on nfc enabled cell phones," in *Near Field Communication (NFC), 2011 3rd International Workshop on*. IEEE, 2011, pp. 77–82.

[4] C. Miller, "Don't stand so close to me: an analysis of the nfc attack surface," *Briefing at BlackHat USA. Las Vegas, NV, USA*, 2012.

[5] Wall of Sheep, "Nfc security awareness project," http://www.wallofsheep.com/pages/nfc-security-awareness-project, 2013.

[6] K. Gold, S. Shetty, and T. Rogers, "A testbed for modeling and detecting attacks on nfc enabled mobile devices," in *Military Communications Conference, MILCOM 2015-2015 IEEE*. IEEE, 2015, pp. 635–640.

[7] IHS Inc., "Nfc-enabled cellphone shipments to soar fourfold in next five years," http://press.ihs.com/press-release/design-supply-chain/nfc-enabled-cellphone-shipments-soar-fourfold-next-five-years, 2014.

[8] Google, "Distribution dashboard — android developers," https://developer.android.com/about/dashboards/, 2018.

[9] L. Du, "An overview of mobile capacitive touch technologies trends," *arXiv preprint arXiv:1612.08227*, 2016.

[10] R. Hattori, "touch panel," http://www.astec.kyushu-u.ac.jp/hat-lab/FPD/TouchPanel.pdf, 2016.

[11] G. Git, "res/values/strings.xml," https://android.googlesource.com/platform/packages/apps/Nfc/+/master/res/values/strings.xml\#73, 2017.

[12] IEEE Standard, "Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2012*, March 2012.

[13] H. W. Klein, "Noise immunity of touchscreen devices," http://www.cypress.com/file/120641/download, 2013.

[14] IDC, "Smartphone os market share," https://www.idc.com/promo/smartphone-market-share/os, 2018.

[15] Mordor Intelligence, "Global near field communication (nfc) market analysis & trends – industry forecast to 2025," https://www.mordorintelligence.com/industry-reports/global-near-field-communication-in-healthcare-market-industry, 2018.

[16] statista, "U.s. proximity mobile payment transaction value 2021 — statistic," https://www.statista.com/statistics/244475/proximity-mobile-payment-transaction-value-in-the-united-states/, 2017.

[17] eMarketer Inc., "emarketer releases new global proximity mobile payment figures," https://retail.emarketer.com/article/emarketer-releases-new-global-proximity-mobile-payment-figures/5a821109ebd4000744ae4118, 2018.

[18] D. F. Kune, J. Backes, S. S. Clark, D. Kramer, M. Reynolds, K. Fu, Y. Kim, and W. Xu, "Ghost talk: Mitigating emi signal injection attacks against analog sensors," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 145–159.

[19] W3C, "Web nfc api security and privacy considerations," https://w3c.github.io/web-nfc/security-privacy.html, 2018.

[20] G. Baldini, J. Löschner, V. Mahieu, R. Neisse, S. Scheer, D. Shaw, and L. Sportiello, "Rfid tag privacy threats and countermeasures: Current status," *Networks*, vol. 149, p. 164, 2005.

[21] A. Juels, "Rfid security and privacy: a research survey," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 381–394, Feb 2006.

[22] A. J. Aviv, K. L. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens." in *Proceedings of the USENIX Workshop on Offensive Technologies (WOOT 2010)*, 2010, pp. 1–7.

[23] F. Maggi, S. Gasparini, and G. Boracchi, "A fast eavesdropping attack against touchscreens," in *Information Assurance and Security (IAS), 2011 7th International Conference on*. IEEE, 2011, pp. 320–325.

[24] Y. Hayashi, N. Homma, M. Miura, T. Aoki, and H. Sone, "A threat for tablet pcs in public space: Remote visualization of screen images using em emanation," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. ACM, 2014, pp. 954–965. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660292

[25] WhichBrowser, "Whichbrowser/parser-php: Browser sniffing gone too far a useragent parser library for php," https://github.com/WhichBrowser/Parser-PHP, 2018.

APPENDIX

A. Smartphone status monitoring system

By carefully analyzing communication between a smartphone and the NFC devices, the system can detect the status of the smartphones, i.e., whether they are locked or unlocked. The key idea of the monitoring system is to leverage the observation that if a smartphone with host card emulation (HCE) enabled is unlocked and receives a `SEL_REQ` command with P2P bit set from a NFC reader. The system periodically probes a smartphone by sending a `SEL_REQ` command using the NFC readers embedded in it. If a HCE-enabled smartphone is brought close to the table top in the unlock state, the smartphone will respond to the probe frame by sending back `SEL_RES` command with the P2P bit set.

Interested readers can download the full code and the collected logs from https://github.com/Tap-and-Ghost/Tap-n-Ghost[10].

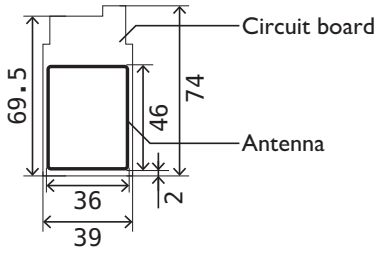

Fig. 14. Diagram of an NFC reader/writer.

[10]We noticed that several models of Android always send back `SEL_RES` command with P2P bit set regardless of whether the smartphone is locked or not, due to vendor customization.

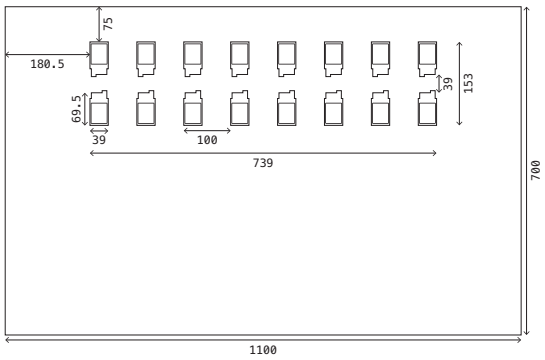

Fig. 15. Layout of NFC reader/writers under a table top.

Fig. 16. Overview of the smartphone status monitoring system inside a table.

Fig. 17. Overview of the smartphone status monitoring system inside a table with table cloth.

B. Supplemental data

We provide supplemental data that are omitted in the main body of the paper due to the space limitation.

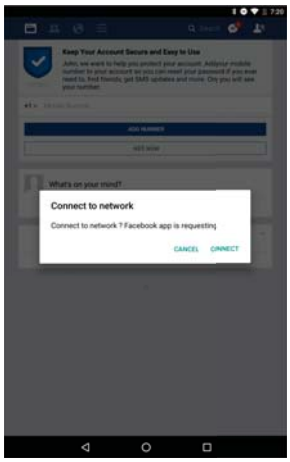

Fig. 18. Wi-Fi connection dialog box (attack using Facebook app).

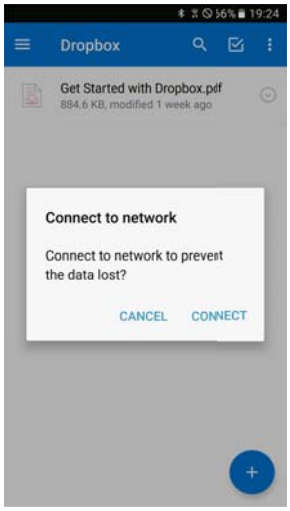

Fig. 19. Wi-Fi connection dialog box (attack using Dropbox app).

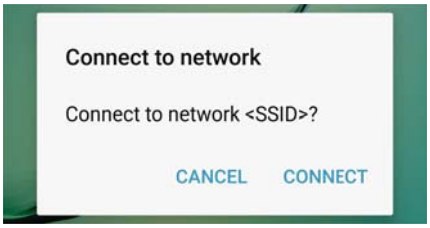

Fig. 20. Wi-Fi connection dialog box (normal).

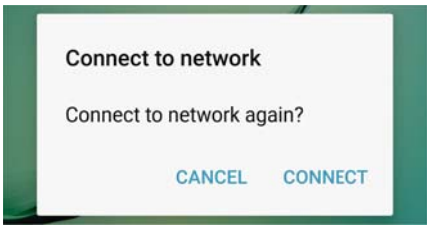

Fig. 21. Wi-Fi connection dialog box (attacked).

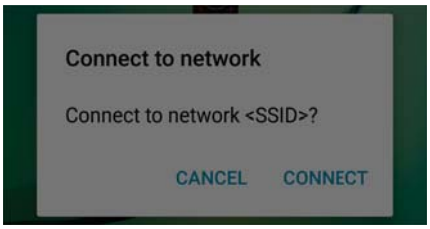

Fig. 22. Wi-Fi connection dialog box (dimmed using Screen Filter app).

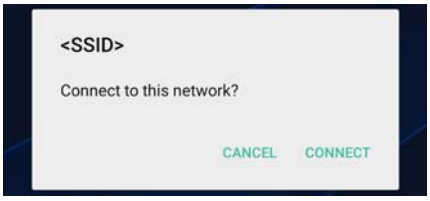

Fig. 23. Wi-Fi connection dialog box (customized for Xperia Z3).

```
<?php
   require 'vendor/autoload.php';
   $result = new WhichBrowser\Parser(getallheaders());
?>

<script>
   var url = (
      "http://attacker.website.com/device_info"
      + "?model=" + "<?php␣echo␣$result->device->model␣?>"
      + "&language=" + navigator.language
      + "&orientation=" + screen.orientation.type
   );
   var xhr = new XMLHttpRequest();
   xhr.open("GET", url, false);
   xhr.send(null);
   location.replace("about:blank");
</script>
```

Fig. 24. An example of snippet code that implements the device fingerprinting. The code makes uses of a third-party library called WhichBrowser [25]. After the device fingerprinting completes, the browser will be redirected to a legitimate website such as `http://www.google.com` or a blank page using the following JavaScript method: "location.replace("about:blank");" which replaces the current URL to the new one and removes the original URL from the session history; i.e., you cannot use the "back" button to return to the original URL. Therefore, a victim will not notice that the device has accessed to a website used for the device fingerprinting.

TABLE V
LIST OF CONFIRMATION MESSAGES INVOKED BY THE WIFICONFIG RECORD.

Type	Title	Message	Positive Button	Negative Button
WI-EN-1	Connect to network	Connect to network <SSID>?	CONNECT	CANCEL
WI-EN-2	Connect	Connect to <SSID>?	YES	NO
WI-EN-3	<SSID>	Connct to this network?	CONNECT	CANCEL

TABLE VI
LIST OF CONFIRMATION MESSAGES INVOKED BY THE BTSSP RECORD

Type	Title	Message	Positive Button	Negative Button
BT-EN-1	—	Are you sure you want to pair the Bluetooth device ?	YES	NO
BT-EN-2	—	Bluetooth pairing requested. Pair?	YES	NO
BT-EN-3	—	Pair with [<name>]?	YES	NO
BT-EN-4	NFC pairing request	Pair with the Bluetooth device ?	Pair	Cancel
BT-EN-5	—	Pair the Bluetooth device ?	YES	NO

TABLE VII
RESULTS OF FEASIBILITY STUDIES.

Device	Manufacture	Android Version	Maximum Reading Distance [cm]	NFC R/W Activated in Factory State	Message Type (Wi-Fi)	Message Type (Bluetooth)
ONETOUCH IDOL 2 S	ALCATEL	4.3	3.0		—	BT-EN-1
Nexus 7	ASUS	6.0.1	4.0	✓	WI-EN-1	BT-EN-1
SAMURAI KIWAMI	FREETEL	5.1	3.0		WI-EN-1	BT-EN-1
ARROWS NX F-05F	FUJITSU	5.0.2	4.0		WI-EN-1	BT-EN-1
Nexus 9	HTC	7.0	4.5	✓	WI-EN-1	BT-EN-1
INFOBAR A02	HTC	4.1.1	2.5		—	BT-EN-1
Ascend P7	HUAWEI	4.4.2	3.5	✓	—	BT-EN-4
TORQUE G02	KYOCERA	5.1	3.5	✓	WI-EN-1	BT-EN-1
TORQUE G01	KYOCERA	4.4.2	3.5	✓	—	BT-EN-1
Nexus 5X	LG	6.0	4.5	✓	WI-EN-1	BT-EN-1
isai vivid	LG	5.1	5.0	✓	WI-EN-2	BT-EN-2
DM-01G	LG	5.0.2	5.0		WI-EN-2	BT-EN-2
ELUGA P	PANASONIC	4.2.2	2.0		—	BT-EN-1
Galaxy S7 edge	SAMSUNG	6.0.1	3.0	✓	WI-EN-1	BT-EN-5
Galaxy S6 edge	SAMSUNG	6.0.1	2.0	✓	WI-EN-1	BT-EN-5
Galaxy S4	SAMSUNG	5.0.1	3.0		WI-EN-1	BT-EN-5
AQUOS ZETA SH-01H	SHARP	5.1.1	3.5	✓	WI-EN-1	BT-EN-1
AQUOS ZETA SH-04F	SHARP	5.0.2	3.5	✓	WI-EN-1	BT-EN-1
AQUOS SERIE	SHARP	5.0.2	3.0	✓	WI-EN-1	BT-EN-1
Xperia XZ	SONY	7.0	3.0	✓	WI-EN-1	BT-EN-3
Xperia Z5	SONY	6.0	3.0	✓	WI-EN-1	BT-EN-3
Xperia Z4	SONY	6.0	4.0	✓	WI-EN-1	BT-EN-3
Xperia Z3	SONY	5.0.2	3.0	✓	WI-EN-3	BT-EN-3
Xperia Z2	SONY	5.0.2	2.5		WI-EN-3	BT-EN-3