NICScatter: Backscatter as a Covert Channel in Mobile Devices

Zhice Yang
SIST, ShanghaiTech University
& CSE, Hong Kong University of Science and Technology
yangzhc@shanghaitech.edu.cn

Qianyi Huang
CSE, Hong Kong University of Science and Technology
qhuangaa@connect.ust.hk

Qian Zhang
CSE, Hong Kong University of Science and Technology
qianzh@cse.ust.hk

ABSTRACT

Today's mobile devices contain sensitive data, which raises concerns about data security. This paper discusses a covert channel threat on existing mobile systems. Through it, malware can wirelessly leak information without making network connections or emitting signals, such as sound, EMR, vibration, etc., that we can feel or are aware of.

The covert channel is built on a communication method that we call NICScatter. NICScatter transmitter malware forces mobile devices, such as mobile phones, tablets or laptops, to reflect surrounding RF signals to covertly convey information. The operation is achieved by controlling the impedance of a device's wireless network interface card (NIC). Importantly, the operation requires no special privileges on current mobile OSs, which allows the malware to stealthily pass sensitive data to an attacker's nearby mobile device, which can then decode the signal and thus effectively gather the guarded data. Our experiments with different mobile devices show that the covert channel can achieve 1.6 bps and transmit as far as 2 meters. In a through-the-wall scenario, it can transmit up to 70 cm.

CCS CONCEPTS

• **Security and privacy → Mobile and wireless security**; • **Networks → Cyber-physical networks**;

KEYWORDS

Backscatter; Commercial Devices; Covert Channel; Wireless

1 INTRODUCTION

Mobile devices, including laptops, smartphones, wearables, etc., have become essential tools in modern life. We rely on them for social activities, document processing as well as health status monitoring. As these devices contain sensitive personal information, various security mechanisms, like firewalls, traffic monitors, and information flow control systems [15], have been developed for mobile devices to prevent unauthorized data leakage.

Nevertheless, our data is not safe enough. Covert channels remain an open threat to data security. A covert channel allows an attacker to leak information from a compromised system even without establishing explicit networked or logical connections. Covert channels exploit media that are usually not treated as data communication channels, and thus they are stealthy and hard for security systems to detect. For example, electromagnetic radiation (EMR) from a display interface [37], EMR from a CPU-to-memory bus [17], the magnetic field from a hard disk [33], etc. are feasible media to establish covert channels. Existing discussions on covert channels usually focus on desktops or servers and cannot be easily extended to mobile situations.

We thus introduce NICScatter, a covert communication method in which a mobile device backscatters surrounding radio frequency (RF) signals with different intensities, surreptitiously leaking out sensitive data from the mobile device's storage. A nearby receiver can then decode the leaked data from the reflected RF signals. Transmitting data through backscattered RF signals is not a new idea, but in conventional wisdom, such a form of communication relies on dedicated hardware to manipulate the impedance of the RF circuits. However, we show that NICScatter can work on a normal commercial mobile device with a wireless transceiver.

Specifically, NICScatter is based on the observation that the impedance of a Wi-Fi Network Interface Card (NIC) varies in different working states, which can be easily modulated via software commands. In this way, like conventional backscatter transmitters, the host mobile device reflects incident RF signals with different intensities, which can be used to covertly convey information to the receiver. Importantly, we can show that in current Linux and Android OSs, the operation of a NICScatter transmitter does not require special privileges. For example, in Linux, NICScatter requires no root privileges. In Android, apps using NICScatter only require a normal permission `CHANGE_WIFI_STATE` [1], which is treated as low risk and will not trigger a message to notify the user during the app installation. These features in current OSs make communication through NICScatter stealthy.

We have prototyped NICScatter transmitter malware in both laptops and smartphones. In addition, we have successfully tested two types of receivers for NICScatter. One is based on Received Signal Strength Indicator (RSSI)/Channel State Information (CSI) measurement ability, which covers many common wireless devices, like laptops and UAVs [30]. This type of receiver can achieve a 0.3bps data transmission rate and up to 0.4m communication range. Another type is based on a software-defined radio (SDR) platform which covers devices with a strong ability to capture wireless signals, like emerging wireless sensing platforms [10, 11, 41]. This type of receiver can achieve a data transmission rate of 1.6bps and a communication range of up to 2m from the transmitter. Through our prototype system, we can demonstrate that our malware can leak data from mobile devices without generating any network

MobiCom '17, October 16-20, 2017, Snowbird, UT, USA.
 ISBN 978-1-4503-4916-1/17/10…$15.00
DOI: https://doi.org/10.1145/3117811.3117814

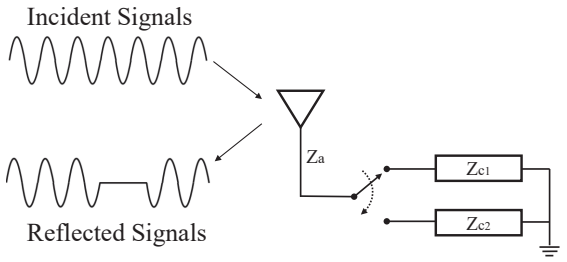


Figure 1: Backscatter Communication. The switch of a transmitter toggles its antenna's load to different impedances. As a result, the RF signals reflected by the antenna have different amplitudes, which are used to convey information, *e.g.*, a stream of encoded data, to the receiver.

traffic or any signals, such as EMR, sound, vibration, heat, etc., that users can feel or be aware of.

We summarize contributions of our work as follows:

1. We introduce a new form of backscatter communication, NICScatter, which uses software control instructions to change the impedance of NICs. In this way, a mobile device can reflect RF signals for communication. NICScatter is the first publicly discussed backscatter design that requires no dedicated hardware and works on today's Wi-Fi NICs.

2. We investigate the vulnerability of current Linux and Android mobile OSs to covertly communicate through NICScatter. By showing that covertly transmitting a stream of encoded data through NICScatter requires no special privileges, we expose the threat of unauthorized information leakage.

3. We evaluate the practicality of implementing a transmitter and receiver for NICScatter. Based on our prototype, we use extensive experiments under realistic settings to validate the feasibility of using NICScatter for covert communication.

The remaining parts of this paper are arranged as follows: we first give the overview for NICScatter in §2. Then we introduce the transmitter and receiver design of NICScatter in §3 and §4, respectively. §5 evaluates the practicality of implementing a covert channel in mobile devices through NICScatter. Related works about covert channels and backscatter communication are presented in §6. We discuss countermeasures, limitations and future work in §7. In §8, we conclude the work.

2 OVERVIEW

This section first gives the background and motivation of the NICScatter design. Then, it introduces a data exfiltration attack on mobile devices based on the NICScatter transmitter.

2.1 Backscatter Communication

Backscatter is a wireless communication method. A backscatter transmitter reflects surrounding RF signals to convey information. It is very power-efficient since the transmitter does not consume power in actively generating RF signals, like conventional RF transmitters do. As a result, it has been widely used in communication scenarios with ultra low power constraints, such as with RFID tags [39].

The working mechanism of backscatter is shown in Figure 1. When an incident RF signal encounters the antenna, a fraction of the signal is reflected/scattered off the antenna. When the chip impedance Z_c is in different states Z_{c1} or Z_{c2}, the amplitude of the reflected signal is also different. This property allows a backscatter transmitter to convey information, *e.g.*, a stream of encoded data, to a receiver. Given the power of the incident signal $P_{incident}$, the power of the backscatter signal $P_{reflected}$ is determined by:

$$P_{reflected} = P_{incident} \cdot \frac{|\rho_1 - \rho_2|^2}{4}, \quad (1)$$

where $\rho_{1,2} = \frac{Z_{c1,c2} - Z_a^*}{Z_{c1,c2} + Z_a} \cdot \frac{|\rho_1 - \rho_2|^2}{4}$ quantifies the efficiency in transforming the received signal into the backscatter signal[27, 34]. Since most Wi-Fi antennas are typically manufactured to $Z_a = 50\Omega$, the reflection efficiency is determined by the chip impedances of the two states. For example, when $Z_{c1} = 0$ and $Z_{c2} = \infty$, $\frac{|\rho_1 - \rho_2|^2}{4}$ reaches its maximum efficiency of 0dB loss.

Model	Off state	Unstable state	On state
AR9380, ant1	$22.5 + j9.2$	$36.3 - j10.0i$	$22.5 + j9.2$
BCM1045, ant1	$12.4 + j14.4$	N/A	$106.9 - j9.8$
Intel5300, ant1	$119.0 + j137.0$	N/A	$80.3 + j66.8$
Intel5300, ant2	$54.3 + j122.7$	N/A	$57.7 + j78.4$

Table 1: NIC Impedance (Ω). Wi-Fi NICs have different impedances in different working states. BCM 1045 and Intel 5300 have different impedances in on and off states. Atheros AR9380 has different impedances in the unstable state when it is switched from the off state to the on state.

2.2 NICScatter: Backscatter through NIC

As consumer mobile devices have no dedicated hardware to control the impedance of the RF circuits, backscatter is typically not considered as a possible communication method for them. NICScatter challenges such conventional wisdom. Its design is based on the interesting property that commercial Wi-Fi NICs have different circuit impedances in different working states.

We measured the impedance of Wi-Fi NICs from some major manufacturers. The measurement was done by connecting the RF port of each NIC to a Roche & Schwarz Vector Network Analyzer [5]. The measurement frequency and power were set to 2.4GHz and -40dBm, respectively. During the measurement, the NIC was switched between on and off states. The impedance readings are recorded in Table 1. We observe that the impedance of all these NICs changes when they are switched from the off state to the on state. BCM 1045 and Intel 5300 hold two different impedances in the two states. The impedance of Atheros AR9380 experiences a temporary change from the off to the on state.

The design of NICScatter is based on this observation. Similar to the impedance switch in Figure 1, NICScatter uses software instructions to switch the commercial Wi-Fi NICs between on and off states. Different circuit impedances in the two states result in different amplitudes of the reflected RF signals by the mobile

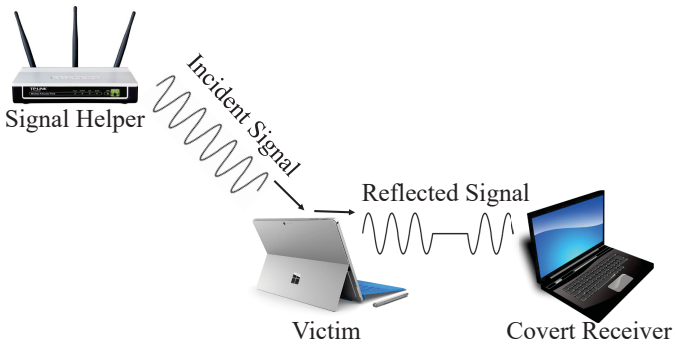


Figure 2: Attack Scenario. The malware in the victim leverages NICScatter to reflect the RF signal from the Signal Helper to covertly leak sensitive information, *e.g.*, a user password, to the Covert Receiver.

device. In this way, NICScatter enables backscatter communication in mobile devices equipped with a Wi-Fi NIC.

As we will show later in §3, the operation of NICScatter does not require root access privileges in Linux systems and only requires a normal access permission in Android systems. The special wireless communication ability combined with the stealthy nature makes NICScatter an ideal choice for a covert channel. In the following subsection, we describe a data exfiltration attack on mobile devices through NICScatter.

2.3 Attack Model

Figure 2 shows the attack scenario. We assume that the mobile devices, *e.g.*, smartphones or laptops, have somehow been compromised by the attacker and have had malware installed with NICScatter capability. The malware has access to certain sensitive data of interest but avoids using the network, because network traffic, as a major information flow, is normally under severe monitoring by many security mechanisms [15]. Moreover, if an application with access to the sensitive data also requires public network connections, it will raise serious attention and awareness of users. Leveraging NICScatter instead of network connections to leak information makes the malware inconspicuous and stealthy.

The covert receiver is positioned close to but isolated from the compromised mobile device; there are no physical or logical connections between them. We assume that the covert receiver can exploit the RF signals generated by a signal helper, *e.g.*, a nearby neutral access point, to enable backscatter from the victim. We also assume that the covert receiver can detect and decode the reflections from the victim. Attacks are likely to occur in public places, like cafes, libraries, transportation stations, *etc.* The attacker uses his/her laptop/AP as the receiver to gather information from neighboring victims. In another case, as modern mobile machines like UAVs [30] also have the capability to receive and decode covert information, they could be abused to gather information from victims in a dense residential environment.

The covert receiver could also be an emerging wireless sensing device, which is designed to perform localization, gesture recognition, *etc.* [10, 11, 41]. Some of these sensing platforms have a strong ability to identify changes in wireless signals, and thus could be used to receive signals reflected by victims. With more and more sensing platforms being developed and accessible, the attacker gains a powerful tool to receive covert information from the victims.

In the examples above, the receiver is not necessarily connected to the Internet. The attacker owns the receiver and collects data from the receiver directly. In a more complicated model, the receivers, such as public APs, may also be compromised by the attacker so that the gathered information is also covertly sent to the remote attacker through the Internet.

3 TRANSMITTER

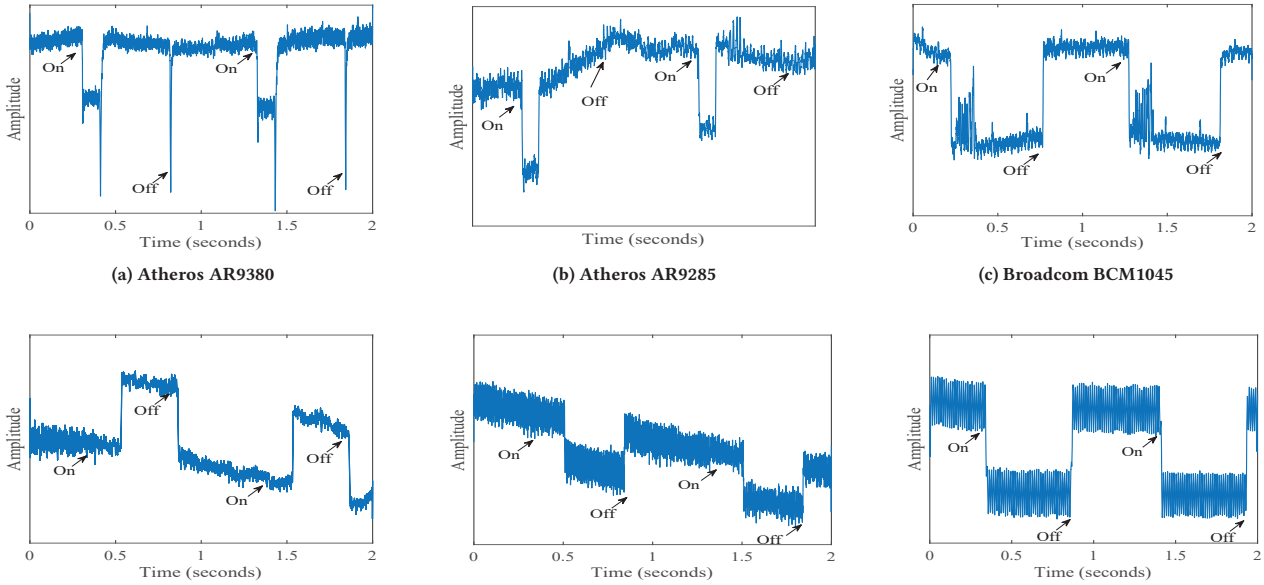
The NICScatter transmitter changes the circuit impedance of the Wi-Fi NIC to reflect RF signals with different intensity for communication. In this section, we introduce its detailed design.

3.1 The Backscatter Properties of Wi-Fi NIC

To exploit the properties of NIC hardware, we conduct experiments to show how the working states affect the reflection. The topology is similar to Figure 2. A USRP is used as the signal helper to generate a sine wave with constant amplitude at 2.4GHz. The antenna of the Wi-Fi NIC of the victim is placed 1m from the signal helper. Another USRP is placed 1m from the victim to capture the reflections. The Wi-Fi NIC is turned on and off every 0.5 seconds. The captured signals are shown in Figure 3. The periodic switching of the NIC impedance results in periodic patterns in the reflected RF signals. Note that once we turn off the signal helper, the periodic patterns disappear immediately at the receiver. Thus, the periodic patterns are generated by the reflections of the Wi-Fi NICs. Our detailed observations are summarized in the following regarding hardware, software and phase diversity:

Hardware Diversity. The impedance changes are quite diverse in different Wi-Fi NICs. According to the waveforms of the reflected signals, NICs can be classified into two types. The impedance of the first type is almost static in the on or off state. Intel 5300 and BCM1045 belong to this type. We term them as *static type*. The impedance of the second type only changes in a short period when turning from on to off or turning from off to on. The corresponding waveform in the reflected RF signals is like a pulse or a bump wave. AR9380 and AR9285 belong to this type. In Figure 3 (a), when the NIC is switched from off to on, it has a square bump lasting for 0.1 seconds. When the NIC is switched from on to off, it has a very short pulse. We term NICs of this type as *pulse type*.

Although AR9380 and AR9285 belong to the same type, and their reflected signals are quite similar, this does not suggest that NICs from the same manufacturer have similar features. Another NIC from Intel (Intel 6300), which is not shown in Figure 3, shows different reflection features from Intel 5300. Intel 6300 is *pulse type*, but Intel 5300 is *static type*. Not surprisingly, NICs of the same model have the same features. We get this conclusion from 4 pieces of Atheros AR9380. Also, different RF ports of the same NIC have similar features, which can be seen from Figure 3 (d) and 3 (e). Therefore, NICs of the same model and RF ports of the same NIC have very similar reflection features, but NICs of different manufacturers and NICs of different models from the same manufacturer do not necessarily have similar features.



(a) Atheros AR9380 (b) Atheros AR9285 (c) Broadcom BCM1045

(d) Intel 5300, First Antenna (e) Intel 5300, Second Antenna (f) Intel 5300 CSI Tool Driver [20]

Figure 3: Waveforms of the Reflected RF Signals. A sine signal at 2.4GHz is reflected by commercial NICs when they are turned on and off every 0.5 seconds. The amplitude changes in the reflected RF signals are caused by the different NIC impedances in different working states. The results are consistent with our impedance readings recorded in Table 1.

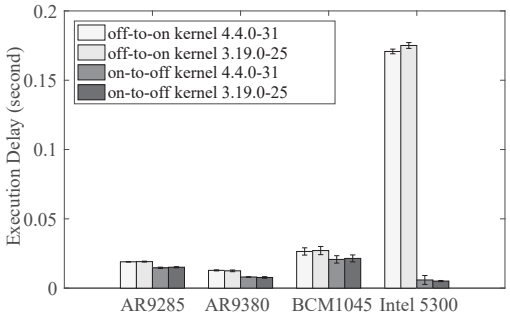


Figure 4: Software on/off Execution Delay of Different NICs. The on/off delay is measured with the same desktop by replacing its WiFi NIC. The software on/off execution time of different NICs is different.

Another important hardware feature is the reflection efficiency. For the ease of comparison, the y-axis of each subfigure in Figure 3 is shifted and scaled to make the waveforms have similar display sizes. However, their absolute amplitudes are quite diverse and are consistent with the impedance readings in Table 1. According to formula (1), the backscatter loss of AR9380, BCM1045 and Intel 5300 is -12dB, -6dB and -18dB, respectively.

Software Diversity. The reflection features also depend on the driver software. Intel 5300 is a NIC of *static type*, but the duty cycle of the reflected RF signals in Figure 3 (d) does not match the duty cycle of the on/off command. This is caused by the execution delay of the NIC driver. To investigate the execution delay of different drivers, we timestamp the instant when we issue commands, *e.g.*, `ifconfig up/down` to turn on/off the NIC, and the instant when the interface is really on/off by checking the `IFF_UP` flag. The execution delay for each NIC is measured 100 times. Results in Figure 4 show that the on-to-off or off-to-on delay is normally around 0.025s. One exception is Intel 5300. Its off-to-on delay is about 10 times higher than that of other NICs. However, in Figure 3 (f), we replace the stocked 5300 driver with the Intel 5300 CSI tool driver [20] and keep other conditions unchanged. The off-to-on delay can be reduced to 0.04s, which is close to that of other NICs. The underlying reason is uncertain for us since its driver uses closed-source firmware. One thing we can conclude is that the reflection features are highly related to the NIC driver.

Phase Diversity. The incident signal and the reflected signal superpose at the receiver's antenna. The final amplitude is related to the location of the receiver. Specifically, if the signal from signal helper is $E_h \sin(\omega t)$ and the signal reflected by the NIC is $E_r \sin(\omega t + \delta)$. The received signal is given by:

$$\sqrt{E_h^2 + E_r^2 + 2E_hE_r \cos\delta}\,\sin(\omega t + \phi), \tag{2}$$

where δ represents the phase difference in the two received signals. The value of δ is determined by the location of the transmitter and the frequency of the RF signal. The two received signals may either superpose constructively or destructively, leading to an increase or decrease in signal amplitudes, respectively. This explains why the direction of the pulses in Figure 3 (d) is different from that of other subfigures.

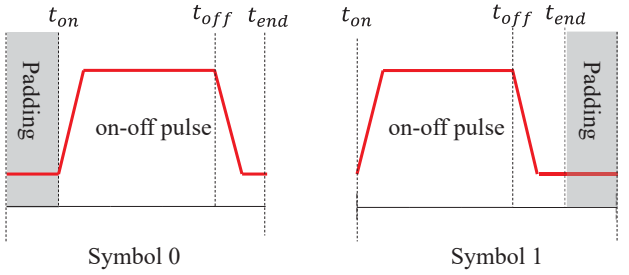


Figure 5: On-off Position Modulation (OOPM). Issuing a pair of off-to-on and on-to-off instructions to the Wi-Fi NIC results in a pulse in the reflected signals. OOPM divides time into equal time slots and uses the location of the on-off pulse in the slot to represent information. In this example, two on-off pulses with different time shifts in the slot are used to represent "0" and "1".

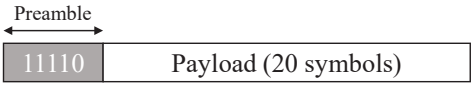


Figure 6: Packet Structure

3.2 Encoding Schemes

Given the reflection properties of different NICs, especially the *pulse type*, it is intuitive to refer to pulse the position modulation (PPM) for the encoding scheme. In PPM, time is divided into equal time slots of duration L, and a single pulse is transmitted during each time slot. The position of the pulse in each time slot determines the symbol being transmitted. If there are M possible position values in one time slot, one symbol represents $\log_2 M$ bits. The corresponding bit rate is $\frac{1}{L}\log_2 M$.

NICScatter uses on-off position modulation (OOPM) to modulate the reflected signal. OOPM is a variant of PPM. Similar to PPM, OOPM divides time into slots and uses the position of a pulse in that slot to represent information. NICScatter generates a pulse in the reflected signal by giving on-off instructions to the NIC. We consider NICs of different types. For NICs of *pulse type*, a single off-to-on or on-to-off instruction is enough to generate a pulse in the reflection. For NICs of *static type*, we need an immediate off-to-on and on-to-off pair to generate a pulse. Note that, in both types, we can use the quick off-to-on and on-to-off pair to generate the pulse in the reflection.

Specifically, NICScatter uses the simplest version of OOPM, where two possible time shifts in one time slot are used to represent one bit. In Figure 5, t_{on} and t_{off} are the time instant when the on and off commands are issued respectively. t_{end} is the finishing time of the on-to-off instructions. $t_{off} - t_{on}$ and $t_{end} - t_{off}$ are the time reserved for the execution delay. The padding period is used to distinguish the position of the on-off pulse in the time slot. The NIC hardware treats the padding period the same as the off state.

We choose t_{on}, t_{off}, t_{end} according to our measurement in Figure 4. Our goal is to unify the transmission scheme among different NICs. So we fix $t_{off} - t_{on}$ to 0.3s and $t_{end} - t_{off}$ to 0.1s

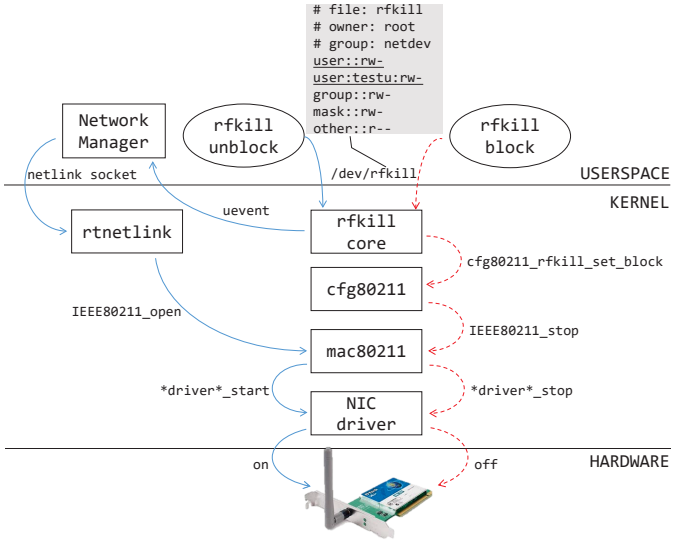


Figure 7: `rfkill` Call Map. `rfkill` is a subsystem in the Linux kernel for switching radio devices. Userspace programs can switch radio devices on and off through the `rfkill` device file at `/dev/rfkill`. The write permission of `/dev/rfkill` is open to the user without root privileges.

to incorporate Intel 5300, which has the largest off-to-on delay of 0.2s. The padding period is set to 0.1s. Thus, the duration of each symbol is 0.5s. Of course, the choice of these parameters can be adjusted to satisfy specific design goals. For example, the symbol period can be much smaller, and rate can be increased by 10× if we exclude Intel 5300.

The packet structure is shown in Figure 6. We use five symbols to transmit the preamble, which contains four consecutive "1"s and one "0". The four "1"s are used for packet detection. The following "0" is used for symbol synchronization, which we will explain later in the receiver design in §4. We use 20 symbols to form the payload, which includes data and the checksum.

Next, we consider implementation issues of the NICScatter transmitter in two popular mobile operating systems: Linux and Android. Our goal is to make the NICScatter transmitter stealthy, and thus we try to avoid the dependency on any sensitive privileges.

3.3 Transmitter in Linux

Linux is widely used on desktops, workstations and laptops. The Ubuntu distribution even has a mobile version for smartphones and tablets [6]. Linux uses privilege control to guarantee the system security. Normally, turning a Wi-Fi NIC on or off is treated as an important system action. To do so, a userspace program must have the root privileges. However, we have found a way to bypass the privilege requirement to enable NICScatter in current Linux systems.

We leverage the vulnerability in the `rfkill` subsystem [4]. `rfkill` is an interface that resides in the Linux kernel to switch radio devices. Drivers for radio devices, such as Wi-Fi, Bluetooth, WAN, GPS, *etc.*, provide interfaces for calls from `rfkill`. Meanwhile,

Algorithm 1 Transmitter

```
1:  procedure TRANSMITTER()
2:      do
3:          t = 0.1s seconds;                    ▷ 5t is one time slot
4:          wait_for_data();
5:          while tx_buffer is not empty do
6:              data ←dequeue(tx_buffer)
7:              if data==1 then
8:                  "unblock wifi"; sleep(3t);
9:                  "block wifi"; sleep(2t);
10:             else
11:                 sleep(t);
12:                 "unblock wifi"; sleep(3t);
13:                 "block wifi"; sleep(t);
14:             end if
15:         end while
16:     while True
17: end procedure
```

`rfkill` responds to events from the user. One typical event is the radio button/airplane mode button on many laptops. The `rfkill` subsystem turns radios on or off according to the voltage of the hardware button. The `rfkill` subsystem also responds to software events. Its interface is exposed to userspace through the device file at `/dev/rfkill`. Writing structured command, *e.g.*, "block wifi", to `/dev/rfkill` turns off the Wi-Fi radio. Note that writing the reverse command "unblock wifi" to `/dev/rfkill` will not turn on the Wi-Fi NIC directly. But as Figure 7 shows, another program, `network_manager`, in the system will automatically turn on the Wi-Fi NIC once the Wi-Fi is unblocked by `rfkill`. By default, `network_manager` is a startup system service. Therefore, applications can write a block or unblock Wi-Fi command to `/dev/rfkill` to turn on or off the Wi-Fi NIC.

We note that the write permission of the `/dev/rfkill` does not require root privileges in many Linux desktop distributions. The vulnerability is caused by Gnome-Bluetooth, which is an application shipped with the Gnome desktop environment. The application installs `61-gnome-bluetooth-rfkill.rules` to release the write permission of `/dev/rfkill` to normal users. The vulnerability is verified in Ubuntu-based distributions with Gnome-Bluetooth version 3.12.0-2 to 3.20.0-1. For newer distributions, the same vulnerability is caused by another application called gnome-settings-daemon [3]. Therefore, any program can use the Algorithm 1 to launch NICScatter transmissions.

3.4 Transmitter in Android

Android is widely used in modem smartphones and wearables. Its transmitter implementation is quite similar to that of Linux. One difference is that the `rfkill` subsystem in Android requires root privileges. To apply NICScatter in non-rooted devices, we propose to use `wifiManager` to control the on-off of the Wi-Fi NIC. Accessing `wifiManager` in an Android app only requires `CHANGE_WIFI_STATE` permission. Note that Android classifies app permissions into two categories. One is the dangerous level, which will explicitly notify users during app installation. The other is normal level, which is treated as low risk and will not notify the user by default. `CHANGE_WIFI_STATE` belongs to the normal level. Malware can take this advantage to hide its NICScatter capability during the installation.

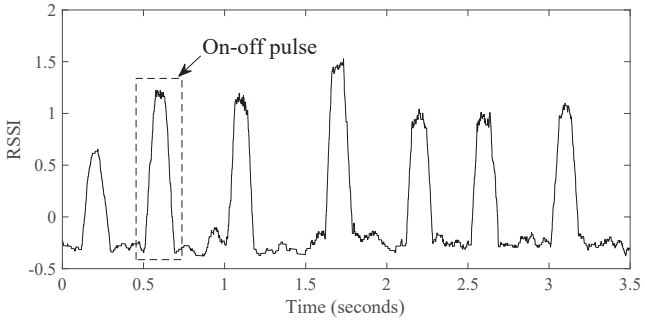


Figure 8: On-off Pulses in a Smoothed RSSI Trace.

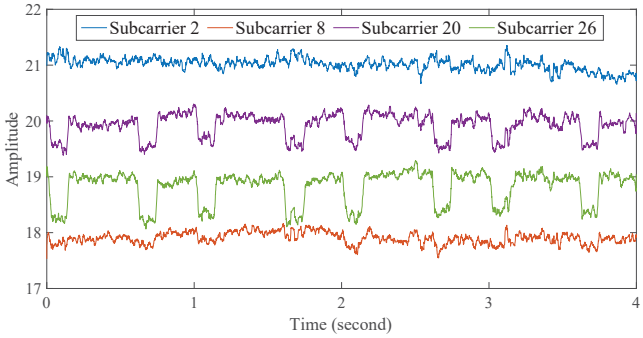


Figure 9: Subcarrier Diversity in CSI Traces.

The transmitter program in Android is similar to Algorithm 1 except the on-off instructions are issued through `wifiManager` rather than `rfkill`. Another difference is the off-to-on delay in Android smartphones is around 1.5 seconds, which is longer than that of Linux laptops. We thus extend the time unit in Algorithm 1 to 0.6 seconds for Android platforms.

4 RECEIVER

This section introduces the NICScatter receiver. Our attack model (§2.3) focuses on two types of receivers. One is based on commercial Wi-Fi NICs, and the other is based on SDRs.

4.1 Receiving with Wi-Fi NIC

To reveal the reflections from the transmitter, NICScatter receiver keeps monitoring Wi-Fi packets from the signal helper. These packets are reflected by the transmitter. As the transmitter is changing its impedance, the signal strength of the reflected packet is also changed accordingly. Like a sampling point in Figure 3, the signal strength of a packet reflected during the on-off pulse is different from that of a packet reflected in the other periods. NICScatter receiver makes use of RSSI and CSI of these Wi-Fi packets to capture

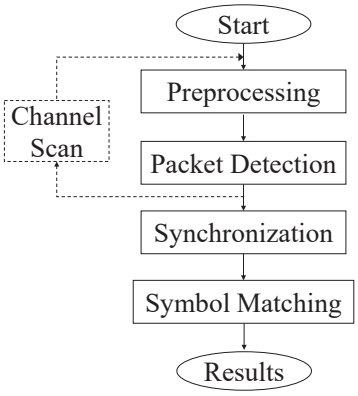


Figure 10: Receiver's Processing Flow.

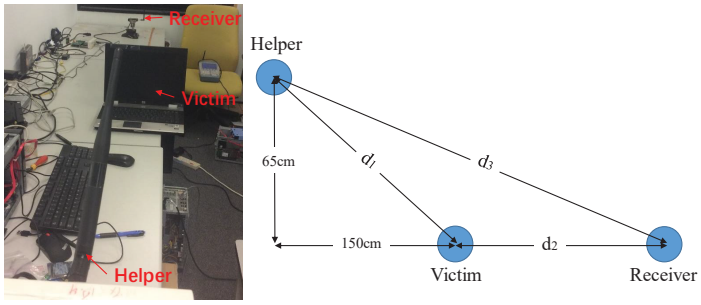


Figure 11: Evaluation Topology

Receiver	Victim	Helper	Result
USRP	Laptop	USRP	§5.1
	Phone		§5.1.7
Laptop	Laptop	Access Point	§5.2

Table 2: Evaluation Outline

the signal changes caused by the transmitter and then decode the information.

Receiving with RSSI. Most Wi-Fi NICs provide the RSSI information of the received packets. We first introduce the decoding method based on RSSI samples obtained from the packets received from the signal helper. An RSSI trace containing on-off pulses is shown in Figure 8. Bits in the trace are extracted with the following four steps:

1. Preprocessing. The recorded RSSI values are interpolated into even sampling intervals. Then, the DC is removed and the trace is smoothed.

2. Packet Detection. Similar to the packet detection process in IEEE 802.11, self-correlation is used to detect the occurrence of the NICScatter packets. Note that every packet starts with a preamble which contains four consecutive "1"s. We choose the duration of one symbol as the self-correlation window. A packet is detected if the self-correlation energy is higher than a threshold and lasts for 3 symbols. As the data payload may also contain four consecutive symbol "1"s, we buffer RSSI values for one packet length and treat every four "1"s as the start of a packet. False positive decoding can be detected through the checksum in the data payload.

3. Synchronization. One challenging problem in decoding PPM-like modulation is to find the exact boundary of the symbols. In our case, since different NICs have different shapes of on-off pulses, the shape of the preamble is unknown to the receiver. Thus, it is not possible to use a known correlation pattern like 802.11 to find the symbol boundary. Thus, the NICScatter receiver determines the symbol boundary through searching. Specifically, if the RSSI values have high self-correlation energy during $[t_1, t_2]$, we choose $T_0 = (t_1 + t_2)/2$, *i.e.*, the center of that period, as the initial boundary. Then, the RSSI samples from T_0 to $T_0 + L$ are treated as symbol "1". According the relation in Figure 5, Symbol "0" is estimated from the samples of the Symbol "1". Note that since the fifth symbol in the preamble is "0", we vary T_0 and fix it as the symbol boundary when the correlation of the estimated symbol "0" and the fifth symbol reaches the highest energy.

4. Symbol Matching. After the previous step, we obtain the template of the symbol "0" and the symbol "1" of the packet. The remaining RSSI samples are then sliced into the unit of one symbol length and correlated with the extracted reference symbol templates. The sliced samples are matched to "0" or "1" according to the correlation energy. The obtained bits are the information carried in the NICScatter packet.

Receiving with CSI. While the above discussion is based on RSSI, it can be easily extended to some Wi-Fi NICs having the CSI capability [20]. Compared with RSSI, CSI contains finer-grained channel information. It quantifies the amplitude and phase shift in subcarriers of the Wi-Fi channel.

One notable difference in the CSI information is the SNR of different subcarriers differs a lot. In Figure 9, subcarriers 20 and 26 have obvious on-off pulses, while the on-off pulses in subcarriers 2 and 8 are not obvious. This can be explained by Equation (2). Due to different wave paths and frequencies, the phase values of the received subcarriers are different, which results in different amplitudes of the received pulses. Therefore, we search the subcarrier of high SNR for decoding. The SNR value can be estimated after the self-correlation. The subcarrier searching process is called "channel scan." The decoding process of the NICScatter receiver with Wi-Fi NIC is summarized in Figure 10.

4.2 Receiving with SDR

We use a USRP N210[7] with XCVR2450[9] as the SDR platform. Using the topology from Figure 2, one USRP is used as the signal helper to generate a sine tone in the 2.4GHz band with constant amplitude. Another USRP is used as the receiver. They use 250kHz as the step in the channel scan. Because the samples of the SDR can be thought of as fine-grained RSSI samples, the decoding process is the same as the one shown in Figure 10.

5 EVALUATION

In this section, we evaluate the communication performance of NICScatter, and its security threat to mobile devices. The evaluation experiments are designed according to the attack scenarios in §2.3.

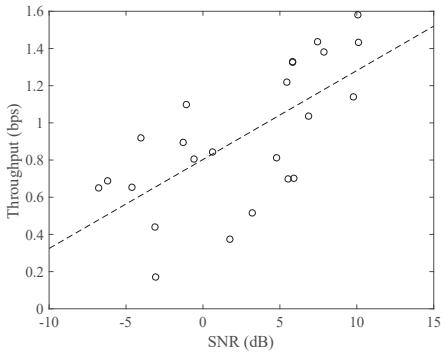


Figure 12: Throughput vs. SNR

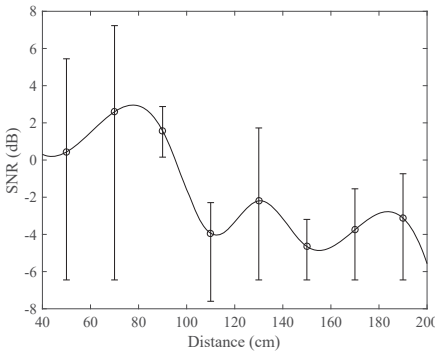


Figure 13: Laptop with Built-in Antenna.

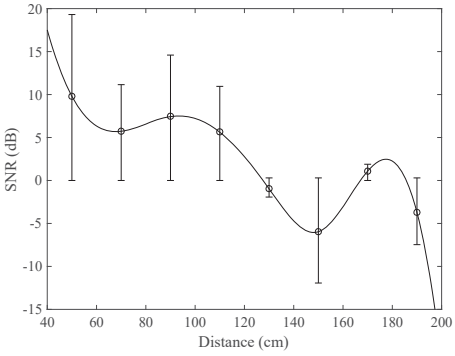


Figure 14: Laptop with External Omni-directional Antenna.

Specifically, we use commercial laptops and mobile phones as NICScatter transmitters, and normal WiFi devices and wireless sensing platforms as NICScatter receivers. Table 2 outlines corresponding evaluation subsections.

5.1 USRP as Receiver

5.1.1 Experiment Setup. The evaluation experiments are conducted in a typical office environment. The topology of the evaluation is shown in Figure 11. The evaluation with USRP as the receiver consists of two attack cases.

In first case, a laptop is the victim and two USRP N210s with XCVR 2450 daughter board are the attacker. This case is used to demonstrate the attack with wireless sensing platforms. Specifically, One USRP is used as the signal helper to generate a sine wave at 2.4GHz. It is equipped with a 5 dBi omni-directional antenna, and the transmit power is 0dBm. Another USRP is used as the receiver. It is equipped with a 2 dBi D-Link omni-directional antenna[2]. Its sampling rate is 250kHz and Rx gain is 45dB. The laptop is HP Elitebook 8530P, which is equipped with a built-in Intel 5300 Wi-Fi NIC.

Settings of the second case (§5.1.7) are the same as the first case, except that the laptop victim is replaced by a Samsung Galaxy S2 smartphone, whose Wi-Fi NIC is Broadcom BCM4330.

5.1.2 Throughput vs. SNR. We first evaluate the throughput under different channel conditions in Figure 12. Each measurement is conducted over a 6 minutes' period, containing 30 packets/600 bits. The payload of each packet is randomly generated. We change the channel condition by moving the receiver to 24 different distances from the victim. As expected, the throughput shows an increasing trend with SNR. Since the throughput is far less than the channel capacity, we can successfully decode packets when the SNR is as low as −5 dB. The maximum achievable throughput is 1.6 bps, which is a little bit lower than the transmission rate, *i.e.*, 2bps. Some bits are not received correctly, mainly due to the following two reasons. First, NICScatter transmits data at very low rates. Thus the mobility of nearby people will dramatically affect the multipath condition during the air time of one bit, resulting in the corruption of the received bits. The second reason is the interference from nearby wireless transmissions. There are about 10 active access points in our evaluation environment. Nearby wireless transmission would distort the received signals, leading to decoding failure.

5.1.3 SNR vs. Distance. The receiver is placed at different distance d_2 from the victim. Figure 13 shows the SNR of the received packets. As expected, the SNR curve has a decreasing trend with the increasing distance. Notice that there are some small fluctuations in the curve. As we described in Section 3, this is because, when the receiver is at different distance to the victim, the reflected signal may either superimpose constructively or destructively with the incident signal. When the receiver is 190 cm away from the victim, the SNR is −3 dB, which is close to the decodable bound in Figure 12. Thus, the effective receiving distance of this setting is about 2 meters.

As we notice the antenna design inside the laptop may affect the performance, we connect the Wi-Fi NIC with an external VERT 2450 3dBi omni-directional antenna[8]. The performance is shown in Figure 14. The overall performance of an external antenna is much better than the built-in antenna, especially when the receiver is close to the victim. This may be caused by the differences in the gain and polarization between the built-in and external antennas.

Note that the received signal is a combination of the backscatter signal and the helper's incident signal. The helper's signal contains no information and is the interference for receiving the backscatter signal. In our test settings, the helper's signal is 40dB to 50dB stronger than the backscatter signal. Since the USRP has 14bit ADC and 84dB dynamic range, the receiver is able to capture the backscatter signal under the presence of the interference. As the interference is a constant sine wave, it can be treated as the DC and easily removed in the received amplitude trace. When the receiver is 2m away from the victim, the power of the backscatter signal is far below the noise floor of the USRP receiver. Thus, given enough ADC dynamic range, increasing the sensitivity level of the USRP receiver is a direct way to increase the communication distance.

5.1.4 Performance of Different Wi-Fi NICs. In Figure 15, we show the performance of different Wi-Fi NICs. As laptops are normally not compatible with different models of Wi-Fi NICs, a desktop with a PCI-E adapter is used as the victim. Different Wi-Fi NICs

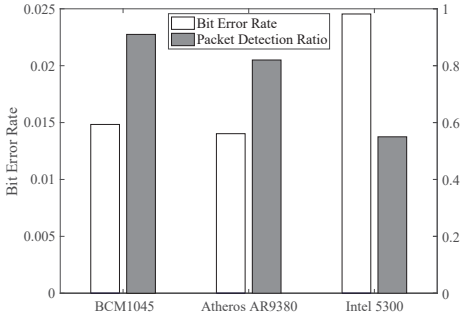


Figure 15: Performance of Different WiFi NIC Models.

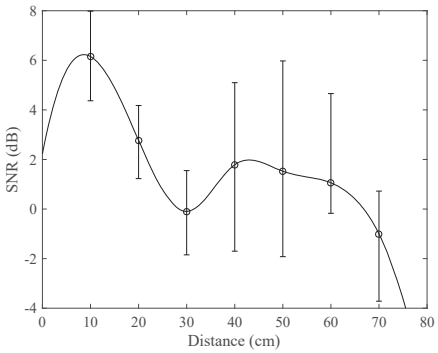


Figure 16: Receiving through the Wall.

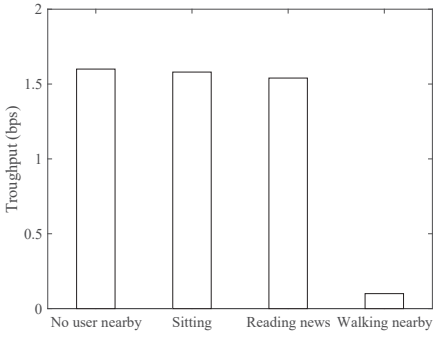


Figure 17: Impact of User Behaviours.

are installed on the PCI-E adapter and connected to a D-Link 2 dBi omni-directional antenna. This experiment covers the products from three major Wi-Fi NIC vendors, including Broadcom, Atheros, and Intel. The distance between the victim and the receiver is fixed to d_2=1m. From Figure 15, we can see that Intel 5300 shows the worst performance among the three. This result matches our impedance measurement in Table 1 and Equation 1. The impedances changes of the Intel 5300 NIC results in the largest backscatter loss among the three. Therefore, if a laptop is equipped with a Wi-Fi NIC from Broadcom or Atheros, its transmission range would likely be larger than 2 meters.

5.1.5 Through-the-Wall Performance. We also test the scenario when the receiver is behind the wall. We place the USRP receiver next to the wall in another room and increase the victim's distance to the wall from 10 cm to 80 cm. The results are shown in Figure 16. When the victim is 70cm away from the wall, the SNR is −1 dB. When the distance further increases to 80 cm, the receiver fails to receive packets. Thus, the communication distance is 70 cm in the through-the-wall scenario.

5.1.6 Throughput with Different User Behaviors. As we describe above, the nearby human movement would change the multipath conditions and affect the throughput. We evaluate the impact with four types of user behaviors: 1) nobody near the laptop; 2) a user sitting before the laptop, with no movement; 3) a user sitting before the laptop and reading news via a web browser; 4) a user walking around near the laptop. The results are shown in Figure 17. The first case is to test the static case, with no human movement; the second case is to see whether human breath/heartbeat would affect the performance; the third case is to show whether small user movement would degrade the performance; the last case is to show the system performance with significant human movement. From the figure, we can see that human breath or small movement would not degrade the system performance. However, significant movement such as walking will significantly degrade the throughput. The attacker may obtain information from the victim when the environment is stable.

5.1.7 Android Phone as Victim. We also evaluate the performance when using an Android phone as the victim. The Tx gain of the signal helper is increased to 10dB. The results are shown in Figure 18. When the receiver is 20 cm away from the victim, the SNR drops below −5 dB. The communication distance is less than that of the laptop. The major reason is the antenna of the mobile phone is more compact and has less gain than that of the laptop. Although the communication distance is limited, the smartphone is still under risks. There are many possible ways for the covert channel to leak information. For example, when making mobile payments, we need to put the smartphone close to a reader, which could be malicious. In another case, a QR code can pretend to be a coupon which in fact just intends to attract people to be close to the receiver.

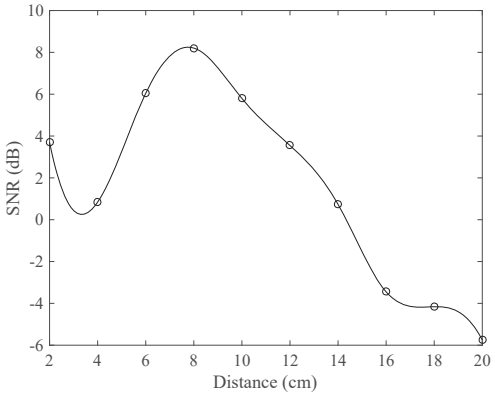


Figure 18: Android Phone as the Victim.

5.2 Laptop as Receiver

5.2.1 Experiment Setup. This scenario is used to demonstrate the attack using commercial Wi-Fi NIC as the receiver. The topology is the same as Figure 11. We uses laptops as the victim and the receiver. Both laptops are equipped with an Intel 5300 Wi-Fi NIC. A TP-Link TL-WDR4360 access point is used as the signal helper. The signal helper generates incident signals by transmitting UDP packets to the receiver. The receiver is running the CSI tool to capture the CSI trace of the received packets. On average, 1000 packets are received within one second. We also assume the attacker

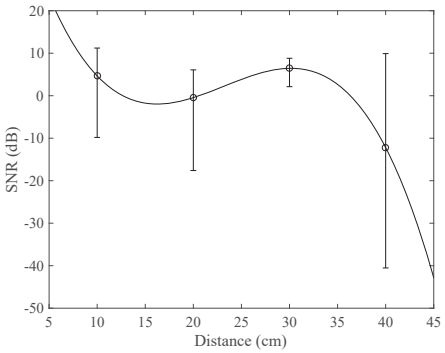


Figure 19: Receving with RSSI.

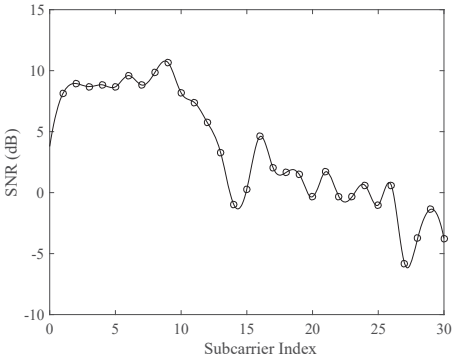


Figure 20: Subcarrier SNR.

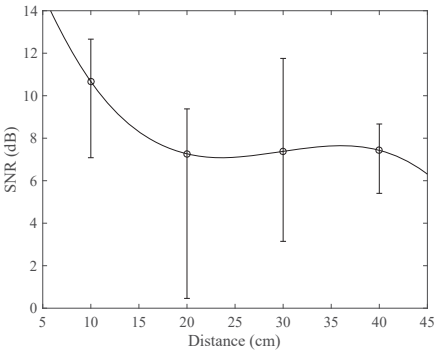


Figure 21: Receiving with CSI.

can improve the receiving ability of its devices. Thus, the receiver's internal antenna is replaced with an external 2dBi D-Link antenna.

5.2.2 Receiving with RSSI. We extract RSSI values from the CSI tool. Note this is not necessary, as RSSI value can also be extracted from the radio tap header in the captured packet. We decode the RSSI trace according to Figure 10, the SNR values are shown in Figure 19. When the victim is 40cm away from the receiver, the SNR drops below −11 dB. At this distance, the receiver can barely decode any packets. Thus, the effective decoding distance is about 30cm.

Compared with USRP, the laptop receiver must be closer to the victim. This is because the CSI tool exposes a smaller number of bits (also smaller dynamic range) than USPRs do. Recall that in our attack scenario the interference is 40dB to 50 dB stronger than the backscatter signal. The variation of the backscatter signal cannot be effectively quantified by the RSSI values. So even the backscatter signal is oversampled in the RSSI trace, the decoding distance is limited.

We note our performance agrees with existing work. The Wi-Fi backscatter transmitter[26] uses a 9dBi[40] directional antenna to reflect the incident signal and achieves 2m receiving distance. As we use a 3dBi antenna in the transmitter, their backscatter loss is 12dB higher than ours. In our evaluation topology, it can be calculated that the SINR at 30cm is 11dB higher than the SINR at 2m, which implies if we replace our transmitter's antenna with a 9dBi antenna, the two systems will have similar performance at 2m.

5.2.3 Receiving with CSI. We go further to analyze the CSI trace. The CSI tool provides channel information for adjacent subcarriers. We observe the same results as in [26] that there are large SNR variations among different subcarriers. Figure 20 shows the SNR for 30 subcarriers when the receiver is 10 cm away from the transmitter. The first to the 10th subcarriers have higher SNR than the remaining ones do. To eliminate the impact of the low SNR subcarriers, we use channel scan to select the high SNR subcarriers for decoding. We show the final results in Figure 21. Comparing Figure 21 with Figure 19, we can learn that CSI has better performance than RSSI. There are two main reasons. First, CSI values are expressed in linear unit and thus have higher resolution than RSSI values, which is expressed in dBm. Second, RSSI is an averaged value over all the subcarriers. Some subcarriers with low SNR may increase the noise in the corresponding RSSI value. With CSI, the communication distance can be larger than 40cm.

Data	bits	Laptop@30cm	USRP@90cm
MAC Address	48	49 sec	54 sec
Plain Password	64	1.09 min	1.20 min
MD5	128	2.18 min	2.40 min
GPS Coordinate	128	2.18 min	2.40 min
SHA1 Hash	160	2.72 min	3.00 min
Disk Encryption Key	256	4.35 min	4.80 min

Table 3: Transmission Time of Sensitive Information

5.3 Attack Summary

According to the above evaluation, we summarize the time needed to transmit the sensitive information in Table 3. In order to increase the effective distance, the attacker can choose to improve the sensitivity of the receiver devices. CSI values are more sensitive than RSSI values. Emerging wireless sensing platforms are more sensitive than commercial Wi-Fi NICs. Attacking small mobile devices, like a smartphone, may require close distance, since these devices use compact antennas, which have a small antenna gain. However, this does not suggest smaller mobile devices are safer. This is because we have more opportunities to carry them and they have more opportunities to be exposed to the risky public environment.

6 RELATED WORK

In this section, we review the related literature from two aspects: covert channels and backscatter communication.

6.1 Covert Channels

Covert channels have been a threat to security systems for a long time. The term was first introduced in 1973 by Lampson [29] to discuss the method through which a program leaks its private data to another process. After that, various covert channel techniques have been developed to bypass security measures to leak information. File locks [16], acoustic signals [36], vibrations [35], CPU

temperature [32], *etc.* have all been used to transmit information between two processes. The concept also has been extended to networked systems, where patterns in network traffic are exploited to convey information [12].

Our work belongs to another type: the covert channel in air-gapped systems, where the victim is isolated from the receiver physically and has no network connections. As early as in the 1950s, the U.S. government became concerned about information leakage in such systems through the EMR emitted by computers' video display units or cables and thus started the TEMPEST program [28]. Over the past few decades, various air-gapped covert channels have been discussed in the public domain. Most of them assume that the victim is a desktop or a server. We compare our work with these covert channels below:

Electromagnetic Radiation. Among various computer components, the video display unit can generate significant EMR [37]. [23] shows how the display content from a laptop or pad can be captured remotely, but the process requires an SDR platform as the receiver. Another work uses a VGA port to generate FM signals [18] for mobile receivers, but not every mobile device has a VGA port. A recent work [33] shows that the EMR generated by the operations of the hard drive can be picked up by a smartphone's magnetic field sensor. However this technique cannot be applied to mobile devices with SSD drives. An interesting work, GSMem [17], leverages the bus operations to generate EMR at the cellular band, but it relies on SSE instructions, which are not available in mobile processors.

Acoustic channels. There are some work exploiting the sound from 18 to 20kHz to establish covert channels [13, 14, 21, 22]. However, one limitation is that the sound in this band can be heard by children or pets, which makes these methods less stealthy.

Thermal emanations. Guri [19] showed that by regulating different workloads on the CPU, they could manipulate the temperature level in the vicinity. These changes can be picked up by the temperature sensors in nearby PCs. However, due to the power constraints, this method cannot be applied to mobile devices.

6.2 Backscatter Communication

Backscatter communication works by reflecting RF signals. To transmit information, a backscatter device switches the load impedance of its antenna, which causes the incident RF signals to be reflected or absorbed. The receiver distinguishes between these two states to decode information. As backscatter devices do not actively generate signal waves, it consumes only microwatts of power. Thus, backscatter devices can be battery-free. RFID technology is an application of backscatter communication.

Recently, Liu *et al.* [31] proposed ambient backscatter, which takes advantage of TV signals in the air and no longer needs a dedicated reader. In a step forward, they [25, 27] enabled tags which can backscatter RF signals to 802.11b packets [25] for Wi-Fi receivers.

The work mostly related to ours is Wi-Fi backscatter [26]. The authors proposed a tag to backscatter Wi-Fi signals to a Wi-Fi reader, which could be an AP or a smartphone. Then, the Wi-Fi reader decodes the data through CSI or RSSI traces. NICScatter uses a Wi-Fi NIC rather than a tag to reflect the Wi-Fi signals. Thus, we have different design goals and constraints. With regard to the performance, their throughput and communication range is better than ours. This is because, as an attacker, we cannot switch the NICs as fast as dedicated tags can and the impedance difference between NICs' on/off states is much smaller than that of dedicated tags.

7 DISCUSSION

In this section, we discuss possible countermeasures against the covert channel enabled by NICScatter. We also discuss limitations and future work of the current NICScatter design.

7.1 Countermeasures

Since NICScatter is based on RF signals, one direct way to block it is shielding, *i.e.*, blocking the incident signal or the reflected signal. However, unlike shielding EMR signals from the display interface or the CPU, shielding a wireless NIC destroys its basic functionalities, which is not desirable for mobile devices.

Another way to disable NICScatter is through the software layer. For Linux systems, cutting off any transition path in Figure 7 will block the NICScatter transmitter. For example, releasing the write permission of a device file, like `/dev/rfkill`, to users without root privileges is usually weird and unnecessary. The Gnome maintainers should be aware of that and find other workarounds for the Bluetooth control application. Another example is to stop `network_manager` from bringing up any Wi-Fi interface automatically. This, however, adds burdens for users. They need to manually turn on Wi-Fi every time when the device leaves the airplane mode. For Android system, hiding the interface for Wi-Fi control may disable many useful applications and sacrifice usability. Thus, the Android society may consider increasing the permission of Wi-Fi control to the dangerous level, so that the permission requirement will be explicitly notified during the app installation. Nevertheless, it is still hard for users to be fully aware of its potential risks.

7.2 Limitations and Future Work

The communication distance of the current NICScatter system may not be sufficient to satisfy the attack from 2m away. In a customized receiver like USRP, if the dynamic range of the ADC is large enough, the distance is limited by the device's sensitivity level. One can adopt a narrow baseband filter and low noise components in the receiver to extend the receiving range. Further, a current NICScatter receiver cannot handle the situation where multiple victims present in the receiving range. The packets from different victims will collide at the receiver and become undecodable. The same problem exists in RFID systems for supporting concurrent transmissions. Similar to the existing work [24], it should be possible to leverage the properties of the oversampled trace and the features of OOPM modulation to decode the packets from multiple concurrent transmitters.

In addition to the covert nature, the insight of NICScatter also implies several interesting points that are worth exploring in the future work. Communicating via backscatter is a promising way to enable network access of low power wireless devices, such as IoTs. While today's discussion mainly focuses on hardware prototypes and PHY layer designs, NICScatter provides a possible way to leverage numerous conventional devices to understand the benefit and the impact of backscatter communication on the network

level. Moreover, NICScatter also inherits the beneficial properties in backscatter signals. For example, as we already have knowledge on localizing RFID tags[38], it is possible to reuse the method to localize conventional wireless devices through NICScatter as well. Further, although we only verify NICScatter on Wi-Fi NICs in this paper, we believe the method we used, *i.e.*, modulating the impedance of the RF circuits through changing its working states, may exist in other RF transceivers, such as cellular, Bluetooth, etc., which may bring other communication opportunities and security issues.

8 CONCLUSION

This paper introduces NICScatter, a backscatter communication method relying on commercial Wi-Fi NICs. NICScatter's design is based on the property that the impedance of a Wi-Fi NIC varies in different working states. NICScatter transmitter thus switches NIC hardware between on and off states to modulate the RF signals reflected by its antenna. The NICScatter receiver extracts information from the transmitter by analyzing the amplitude of the reflected signals. We show that even an ordinary wireless device with RSSI/CSI measurement ability is capable of the decoding process.

As the communication form of NICScatter hasn't caused enough attention in current mobile systems, we show there are vulnerabilities to exploit NICScatter as a covert way to leak information. Our experiments show that the covert channel on a laptop can reach 2 meters away and it can even transmit through a wall. We, therefore, demonstrate a realistic threat of data exfiltration attack on mobile devices.

ACKNOWLEDGEMENT

We would like to thank our Shepherd Fadel Adib, and the anonymous MobiCom reviewers for their constructive comments. We also thank Ted Spaeth for help with proofreading and Shanpu Shen for help with impedance measurement. This work was supported in part by the 973 Project under Grant 2013CB329006, in part by the RGC under Contract CERG 16212714, 16203215, Contract ITS/143/16FP-A, and Contract R8015, in part by the Huawei-HKUST Joint Laboratory (by the IoT WiFi Key Technologies), in part by the ShanghaiTech University Startup Fund under Grant F-0203-17-010.

REFERENCES

[1] Android permissions. https://developer.android.com/reference/android/Manifest.permission.html.
[2] D-link 2dbi antenna. https://www.amazon.com/D-Link-2dbi-Antenna-Wireless-Router/dp/B002XENHB0.
[3] gnome-bluetooth source code. http://sources.debian.net/src/gnome-bluetooth/.
[4] rfkill. https://www.kernel.org/doc/Documentation/rfkill.txt.
[5] R&S®ZVB vector network analyzers. https://www.rohde-schwarz.com/us/product/zvb-productstartpage_63493-7990.html.
[6] Ubuntu mobile. https://www.ubuntu.com/mobile.
[7] Usrp n210. https://www.ettus.com/product/details/UN210-KIT.
[8] Vert 2450 3dbi antenna. https://www.ettus.com/product/details/VERT2450.
[9] Xcvr2450. https://kb.ettus.com/XCVR2450.
[10] F. Adib, Z. Kabelac, D. Katabi, and R. C. Miller. 3d tracking via body radio reflections. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, pages 317–329. USENIX Association, 2014.
[11] F. Adib and D. Katabi. See through walls with wifi! *SIGCOMM Comput. Commun. Rev.*, 43(4):75–86, Aug. 2013.
[12] S. Cabuk, C. E. Brodley, and C. Shields. Ip covert timing channels: design and detection. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 178–187. ACM, 2004.
[13] B. Carrara and C. Adams. On acoustic covert channels between air-gapped systems. In *International Symposium on Foundations and Practice of Security*, pages 3–16. Springer, 2014.
[14] L. Deshotels. Inaudible sound as a covert channel in mobile devices. In *WOOT*, 2014.
[15] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):5, 2014.
[16] V. D. Gligor. *A guide to understanding covert channel analysis of trusted systems*. The Center, 1994.
[17] M. Guri, A. Kachlon, O. Hasson, G. Kedma, Y. Mirsky, and Y. Elovici. Gsmem: Data exfiltration from air-gapped computers over gsm frequencies. In *USENIX Security*, pages 849–864, 2015.
[18] M. Guri, G. Kedma, A. Kachlon, and Y. Elovici. Airhopper: Bridging the air-gap between isolated networks and mobile phones using radio frequencies. In *MALWARE*, pages 58–67. IEEE, 2014.
[19] M. Guri, M. Monitz, Y. Mirski, and Y. Elovici. Bitwhisper: Covert signaling channel between air-gapped computers using thermal manipulations. In *Computer Security Foundations Symposium*, pages 276–289. IEEE, 2015.
[20] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. In *ACM SIGCOMM Computer Communication Review*, volume 40, pages 159–170. ACM, 2010.
[21] M. Hanspach and M. Goetz. On covert acoustical mesh networks in air. *Journal of Communications*, 8(11), 2013.
[22] M. Hanspach and M. Goetz. Recent developments in covert acoustical communications. In *Sicherheit*, pages 243–254. Citeseer, 2014.
[23] 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 CCS*, pages 954–965. ACM, 2014.
[24] P. Hu, P. Zhang, and D. Ganesan. Laissez-faire: Fully asymmetric backscatter communication. *acm special interest group on data communication*, 45(4):255–267, 2015.
[25] V. Iyer, V. Talla, B. Kellogg, S. Gollakota, and J. Smith. Inter-technology backscatter: Towards internet connectivity for implanted devices. In *SIGCOMM*, pages 356–369. ACM, 2016.
[26] B. Kellogg, A. Parks, S. Gollakota, J. R. Smith, and D. Wetherall. Wi-fi backscatter: Internet connectivity for rf-powered devices. *ACM SIGCOMM Computer Communication Review*, 44(4):607–618, 2015.
[27] B. Kellogg, V. Talla, S. Gollakota, and J. R. Smith. Passive wi-fi: Bringing low power to wi-fi transmissions. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 151–164. USENIX Association, 2016.
[28] M. G. Kuhn and R. J. Anderson. Soft tempest: Hidden data transmission using electromagnetic emanations. In *International Workshop on Information Hiding*, pages 124–142. Springer, 1998.
[29] B. W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.
[30] B. Li, S. Zhang, and S. Shen. Csi-based wifi-inertial state estimation.
[31] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith. Ambient backscatter: wireless communication out of thin air. In *SIGCOMM*, pages 39–50. ACM, 2013.
[32] R. J. Masti, D. Rai, A. Ranganathan, C. Müller, L. Thiele, and S. Capkun. Thermal covert channels on multi-core platforms. In *USENIX Security*, pages 865–880, 2015.
[33] N. Matyunin, J. Szefer, S. Biedermann, and S. Katzenbeisser. Covert channels using mobile device's magnetic field sensors. In *Asia and South Pacific Design Automation Conference*, pages 525–532. IEEE, 2016.
[34] P. V. Nikitin and K. V. S. Rao. Antennas and propagation in uhf rfid systems. In *2008 IEEE International Conference on RFID*, pages 277–288, April 2008.
[35] E. Novak, Y. Tang, Z. Hao, Q. Li, and Y. Zhang. Physical media covert channels on smart mobile devices. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 367–378. ACM, 2015.
[36] R. Schlegel, K. Zhang, X.-y. Zhou, M. Intwala, A. Kapadia, and X. Wang. Soundcomber: a stealthy and context-aware sound trojan for smartphones. In *NDSS*, volume 11, pages 17–33, 2011.
[37] W. Van Eck. Electromagnetic radiation from video display units: An eavesdropping risk? *Computers & Security*, 4(4):269–286, 1985.
[38] J. Wang and D. Katabi. Dude, where's my card?: Rfid positioning that works with multipath and non-line of sight. *acm special interest group on data communication*, 43(4):51–62, 2013.
[39] R. Want. An introduction to rfid technology. *IEEE pervasive computing*, 5(1):25–33, 2006.
[40] P. Zhang, M. Rostami, P. Hu, and D. Ganesan. Enabling practical backscatter communication for on-body sensors. pages 370–383, 2016.
[41] M. Zhao, F. Adib, and D. Katabi. Emotion recognition using wireless signals. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pages 95–108. ACM, 2016.