USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

# SNI5GECT: A Practical Approach to Inject aNRchy into 5G NR

Shijie Luo, Matheus Garbelini, Sudipta Chattopadhyay, and Jianying Zhou, *Singapore University of Technology and Design*

https://www.usenix.org/conference/usenixsecurity25/presentation/luo-shijie

This paper is included in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.

# SNI5GECT: A Practical Approach to Inject aNRchy into 5G NR

Shijie Luo, Matheus E. Garbelini, Sudipta Chattopadhyay, and Jianying Zhou
*Singapore University of Technology and Design*

## Abstract

In this paper, we propose and design SNI5GECT– a framework that sniffs messages from pre-authentication 5G communication in real-time and injects targeted attack payload in downlink communication towards the UE. As opposed to using a rogue base station which limits the practicality of many 5G attacks, SNI5GECT acts as a third-party in the communication, silently sniffs messages, and tracks the protocol state by decoding the sniffed messages during the UE attach procedure. The state information is then used to inject targeted attack payload in downlink communication. We have implemented SNI5GECT and evaluated it with five 5G enabled UE devices and with both open-source (srsRAN) and commercial (Effnet) base stations (gNBs). Our evaluation reveals that SNI5GECT obtains over 80% accuracy in uplink and downlink sniffing, and successfully injects messages at an arbitrary protocol state with a 70%-90% success rate up to 20*m* of distance between UE and SNI5GECT. We further evaluate SNI5GECT to launch a variety of attacks that crash the UE, downgrade the connection to lower generation or extract the UE identity with an attack success rate often over 70% with known UE distance. Finally, we discover a new multi-stage, downgrade attack leveraging the SNI5GECT framework. The risk of this attack has been acknowledged by GSMA and a coordinated vulnerability disclosure (CVD) identity has been assigned. Thus, SNI5GECT is a practical and complementary tool for evaluating current and new 5G attacks in the wild.

## 1 Introduction

The fifth generation of mobile networks 5G New Radio (5G NR) holds the potential to influence several key sectors e.g., smart home, industry control, healthcare, and robotics due to its high speed and enhanced security control. Despite the promising features offered by 5G NR, several design and implementation vulnerabilities have already been reported [9, 15, 18, 30]. While such vulnerabilities undermine the trust in 5G network, it is necessary to understand their real-world impact for accurate risk assessment. Therefore, it is crucial to design frameworks and capabilities for investigating the limits and impact of 5G NR attacks, for example, when attackers may be constrained by practical real-world limitations.

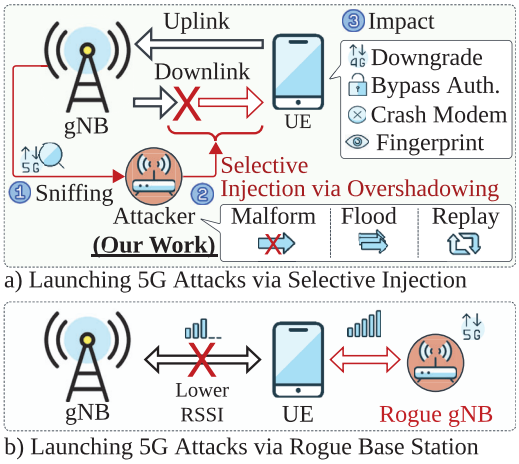


Figure 1: An Illustration of SNI5GECT attack model

In this paper, we propose and design SNI5GECT– a foundational framework for launching pre-authentication attacks, capable of sniffing both downlink and uplink 5G traffic over-the-air. Moreover, SNI5GECT embodies capabilities to inject downlink messages at the correct timing (e.g., after a specific protocol state), causing the User Equipment (UE) to accept the message. Such injected messages can be targeted attack payloads, resulting in the UE, for example, to crash or to downgrade. Figure 1(a) illustrates the attack model targeted by the SNI5GECT framework. As opposed to *heavily intrusive* approaches such as employing rogue gNB [9, 30] (see Figure 1(b)), SNI5GECT passively sniffs messages during the initial connection process, decodes the message content, and leverages the decoded message content to later decide to inject targeted attack payloads. As opposed to prior work [19], SNI5GECT sniffs and decodes both control-plane and user-plane 5G NR messages in real-time (i.e., without requiring of-

fline analysis) and systematically injects attack payload based on protocol state. This makes SNI5GECT suitable to launch fingerprinting, denial-of-service, and downgrade attacks on targets that require injection of messages under different communication states.

*To the best of our knowledge,* SNI5GECT *is the first framework that empowers researchers with both over-the-air sniffing and stateful injection capabilities, without requiring a rogue gNB*. This is particularly important, as launching attacks via a rogue gNB requires use of bespoke techniques to force a UE to connect to the rogue gNB. This adds complexity and extra requirements to the setup, thus, increasing the chance for an attack to fail. Furthermore, a rogue gNB can be easily detected by intrusion detection systems. This is because of the broadcast messages (Master Information Block (MIB) and System Information Block (SIB)) being transmitted all the time from the rogue gNB within the coverage area.

We argue that both real-time sniffing and message injection capabilities towards the UE communication, as embodied in the SNI5GECT framework, are essential for a 5G security tool. Concretely, these capabilities enable arbitrary attack scenarios that are stateful in nature: *(i)* requires sniffing and extracting information from a message, and *(ii)* subsequently, reusing such information and making decisions whether to inject a generated message in multiple stages. This is with the aim of exploiting the implementation or protocol-level vulnerabilities in the UE (e.g., fingerprinting, security bypass, downgrade, etc). For example, launching active fingerprinting attacks requires the injection of a message (`Identity Request`) followed by sniffing of a specific message (`Identity Response`) to successfully extract unique user information such as Subscription Concealed Identifier (SUCI). Moreover, several attacks demand messages to be injected after the *Radio Resource Control (RRC) attach* procedure [9, 15], such as attacks involving Non-access stratum (NAS) messages (i.e., `Registration Reject`). In general, due to the dynamic nature of 5G protocols, the stateful message injection capability of SNI5GECT is essential for practically launching the most complex attacks on 5G UEs.

Prior works on 5G security testing requires use of rogue gNB [9, 15, 30], do not involve comprehensive sniffing of uplink/downlink messages [19], and fail to have fine-grained, arbitrary and stateful message injection capabilities anywhere during the 5G connection process [18, 32]. While the AdaptOver framework [6] performs both message sniffing and injection for 4G networks, such work focuses only on one-shot attacks that do not need to track responses from UEs, involve high latency (≈50ms) and is not directly applicable to 5G NR system. Moreover, the lack of an open framework makes it impossible to potentially extend AdaptOver. In contrast to the aforementioned works, SNI5GECT provides an open framework to sniff pre-authentication uplink/downlink messages in 5G network, and it allows arbitrary message injection by carefully monitoring the states from the sniffed and decoded packets. This, in turn, facilitates orchestration and evaluation of stateful and arbitrary attacks during the 5G connection process in a realistic and non-intrusive fashion.

After providing a brief background and overview of SNI5GECT framework (Section 2), we make the following contributions in the paper:

1. We present the design of SNI5GECT– an open 5G framework that embodies both sniffing and downlink injection capabilities in a stateful fashion (Section 3).

2. We implement SNI5GECT on top of `srsRAN 4G` [26] and `WDissector` library [9]. Using USRP b210, we evaluate its sniffing and stateful injection capabilities with four state-of-the-art smartphones employing different 5G baseband modems. Our evaluation revealed that SNI5GECT obtains $>$ 80% of accuracy when sniffing both downlink and uplink messages up to 20m from the UE. Furthermore, injecting messages after a variety of states show a success rate between 70%-90% (Section 5).

3. We leverage SNI5GECT to launch a variety of pre-authentication attacks where UEs are located at known distance: attacks that do ignore the UE's response (one-shot) [9], attacks that involve waiting for a response from the UE (SUCI catcher), and attacks that involve multiple stages, leveraging multiple sniffing and injection steps to be successful. We show that SNI5GECT achieves an attack success rate of up to 100% for one-shot attacks and up to 93% for both multi-stage attacks and attacks expecting response. We also show that some existing attacks, employing a rogue gNB, may not achieve the same impact when exploited in the wild (Section 5).

4. We have discovered a new multi-stage, downgrade attack using SNI5GECT and have reported this finding responsibly to GSMA. A CVD identifier `CVD-2024-0096` is assigned to acknowledge the associated risk (Section 5).

5. Finally, we demonstrate the generalization of SNI5GECT by evaluating UE attacks when running either Effnet commercial gNB [1] or `srsRAN` open-source gNB. We observe that SNI5GECT achieves similar attack success rates against both alternatives (Section 5).

After discussing the related work (Section 6) and the limitations of our framework (Section 7), we conclude in Section 8.

## 2 Background and Overview

### 2.1 Background on 5G Communication

The top part of Figure 2 shows the spectrogram graph of different signals in 5G, as we introduce in the following sections.

**Cell Search & Synchronization:** In 5G communication, downlink/uplink transmissions are organized into *ten subframes* (a *frame*), which are further divided into slots based on

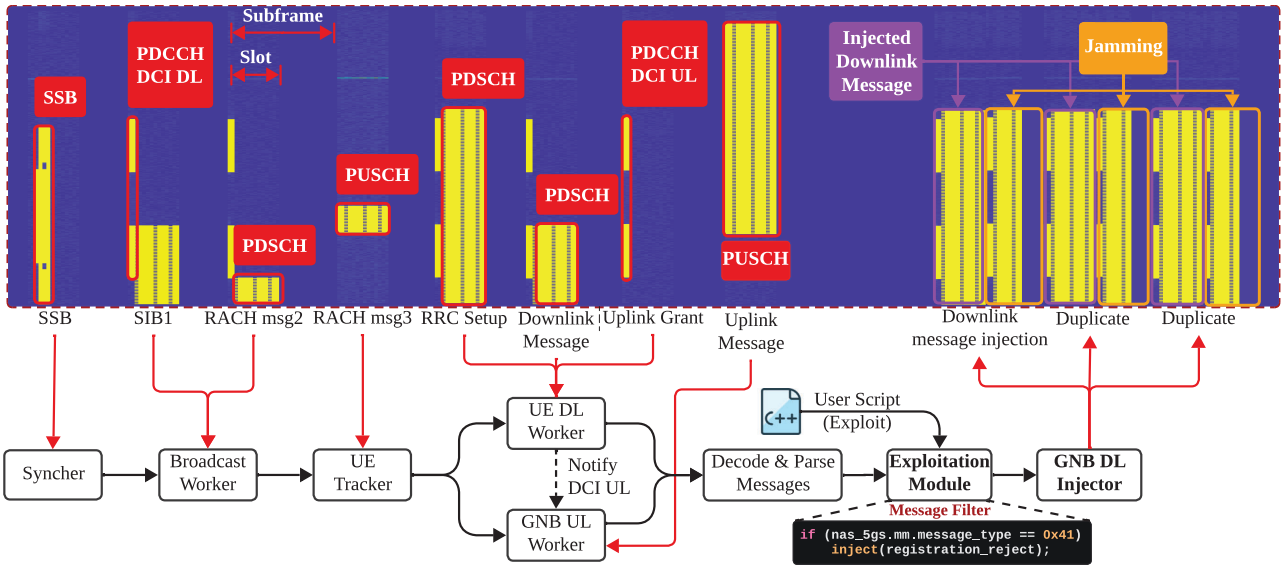


Figure 2: Illustration of SNI5GECT components

subcarrier spacing. The gNB periodically broadcasts the Synchronization Signal Block (SSB) containing three signals: Primary Synchronization Signal (PSS), Secondary Synchronization Signal (SSS), and Physical Broadcast Channel (PBCH). By performing correlation with the predefined PSS and SSS sequences, the UE can synchronize its timing and frequency offsets with the gNB and then decode the PBCH. Then, the UE extracts Master Information Block (MIB) payload from the PBCH, which contains essential cell information, cell-barred status, SSB frequency offset, and other parameters that inform the UE how to decode further signals from the gNB.

**Initial Cell Configuration:** In addition to the MIB, the gNB broadcasts the System Information Block Type 1 (SIB1) on the Physical Downlink Shared Channel (PDSCH). SIB1 contains essential information, such as common uplink and downlink configurations and random access parameters, enabling UEs to connect and communicate with the network.

**PDCCH Search:** Physical Downlink Control Channel (PDCCH) plays a crucial role in delivering control information to the UE. The UE uses configurations from the Control Resource Set (CORESET) and the search space to iteratively search for Downlink Control Information (DCI) within the PDCCH. The DCI informs modulation, and coding schemes, resource allocation, and scheduling information. These indicate *how* to handle and *where* to locate Orthogonal Frequency Division Multiplexing (OFDM) symbols in the Physical Downlink Shared Channel (PDSCH) and the Physical Uplink Shared Channel (PUSCH).

**PDSCH & PUSCH Decoding:** The decoding of PDSCH and PUSCH is critical for the data transmission in a 5G system. Both PDSCH and PUSCH rely on the DCI carried by PDCCH to determine e.g., the allocated resources, modulation, and coding scheme (MCS) for decoding. Using such information, the UE extracts and decodes the data from the specified physical resource blocks (PRBs) and OFDM symbols. For SNI5GECT, we leverage the configuration details obtained from the DCI to decode messages transmitted over the PDSCH and PUSCH between the UE and the gNB. Moreover, if we intend to send a message to a target UE, then we must encode the message into the PDSCH and include a DCI message in the PDCCH to indicate the scheduling information. Finally, PDCCH and PDSCH In-phase and Quadrature (IQ) samples are placed into the appropriate slot and transmitted.

**Random Access:** The UE uses Random Access procedure (RACH) to establish connection with the gNB. First, the UE transmits a `Random Access Preamble` over the Physical Random Access Channel (PRACH). Upon receiving the preamble, the gNB responds with a `Random Access Response` (RAR) message on the PDSCH channel. This message includes key information such as a Temporary Cell Radio Network Temporary Identifier (TC-RNTI), and an uplink grant for the subsequent transmission. By monitoring RACH messages, we can detect new UEs attempting to connect to the network. Following RAR, the UE uses the allocated uplink resources to transmit `RACH Message 3` on PUSCH. This message typically carries `RRC Setup Request`, which includes the contention resolution identity. In response, the gNB transmits `RACH Message 4` on the PDSCH to resolve contention. Once the UE processes and acknowledges these messages, it establishes the RRC connection with the gNB.

## 2.2 Overview of SNI5GECT

SNI5GECT includes three key entities: *(i)* the legitimate User Equipment (UE), *(ii)* the legitimate 5G gNB, and *(iii)* the

attacker. The legitimate UE communicates directly with the gNB, while the attacker initially operates as a passive sniffer positioned between them (see Figure 1).

**Attacker Model:** SNI5GECT concentrates on attacks occurring before the authentication procedure to highlight the impact of vulnerabilities triggered by unencrypted messages. Since messages exchanged between the gNB and the UE are not encrypted before the security context is established (pre-authentication state), an attacker does not require knowledge of the UE's credentials to sniff uplink/downlink nor to inject messages without integrity protection throughout the UE connection procedure. For example, an attacker can exploit the short UE communication window that spans from the RACH process until the NAS security context is established. Such an attacker actively listens for any RAR message from the gNB, which provides the RNTI to decode further UE messages. While this approach requires SNI5GECT to sniff the UE starting from the 5G registration procedure, it is highly practical since many common events can cause a phone to lose signal, resulting the UE to perform re-connection: Users (i) entering and leaving a lift which temporarily breaks mobile connectivity, (ii) arrivals in the airport (toggling of airplane mode), (iii) passing through a long tunnel or an underground parking garage where signal is completely lost. Consequently, an attacker located nearby such places could easily exploit the UE communication window even without using selective 5G jamming techniques.

Nonetheless, injection needs to occur at a precise moment, starting from the RAR procedure, such that the UE fails to distinguish between messages originating from the legitimate gNB and the attacker. This opens opportunities attacking UEs without using a rogue gNB that needs exclusive communication with the UE and knowledge of its credentials. Specifically, our attacker mimics a *partial* Dolev–Yao adversary, which *(i)* is capable of eavesdropping on pre-authentication unencrypted messages, *(ii)* it can inject, replay or modify messages towards the downlink communication channel of an UE at a known distance, and *(iii)* fingerprint and subsequently track the UE by passive (sniffing) or by active means (injection).

**Workflow:** SNI5GECT starts by synchronizing with the legitimate gNB via the *Syncher* component (see Section 3.1 for details). Then, the time-corrected subframes are passed to the *Broadcast Worker* for search and extraction of SIB1 parameters. As discussed in Section 2.1, SIB1 includes several crucial configurations e.g., common uplink and downlink configurations. Thus, the *Broadcast Worker* decodes the SIB1 and applies the decoded settings for subsequent communication tracking (see Section 3.2). Then, the *Broadcast Worker* also detects whenever a new UE approaches the 5G gNB and spawns an instance of the *UETracker* component.

For each *UETracker* instance, every subframe received from the *Syncher* is passed to its associated *UE DL Worker* (see Section 3.3.1) and *GNB UL Worker* (see Section 3.3.2). The *UE DL Worker* handles downlink control information (DCI) searches and PDSCH channel decoding, while the *GNB UL Worker* performs PUSCH channel decoding. Once messages are decoded by the *UE DL Worker* and *GNB UL Worker*, they are leveraged to identify the protocol/communication state and depending on the protocol state, a crafted attack payload might be generated. The attack payload is finally passed to the *GNB DL Injector* for encoding and is injected into the UE using a Software-defined radio (SDR) (see Section 3.3.3).

## 3 SNI5GECT Design

In the following, we discuss the two main capabilities of SNI5GECT (Figure 2): ***sniffing*** over-the-air messages and ***injecting*** downlink messages from the gNB to the UE.

### 3.1 Cell Synchronization & Sniffing

This component (named "Syncher") serves as the entry point for activating the sniffer (see Figure 2). Its primary role is to synchronize with the target 5G cell and retrieve the basic information contained in the MIB. To achieve this, the *Syncher* handles challenging and time-sensitive tasks at the 5G physical layer such as maintaining real-time synchronization and distributing subframes to other components. This is a critical requirement for 5G since missing a single slot from either UE or gNB can result in the loss of critical messages, potentially leading to missed opportunities for injecting attack payload to the UE at a precise time. To address these challenges, the *Syncher* is designed to be operated in a thread with the highest scheduling priority. This ensures real-time performance and minimizes latency, enabling SNI5GECT to process the signals and decode the messages in real-time.

Initially, to detect the target cell, the *Syncher* follows an approach similar to a UE device. In particular, after initialization, *Syncher* performs a cell search using predefined configurations to detect the target cell. Specifically, *Syncher* searches for the Synchronization Signal Blocks (SSB) at the specified SSB frequency, processing one subframe at a time. It computes the timing offset, and Carrier Frequency Offset (CFO) of the received subframe. This is then used to extract the Resource Elements (REs) at the correct offsets and decode the PBCH for retrieving MIB.

The cell search step is successful only if the MIB passes CRC verification; otherwise, *Syncher* continues processing subsequent subframes until successful. Once the MIB is extracted, the configurations from the MIB are distributed to other components, such as the *Broadcast Worker* for SIB1 decoding (see Figure 2). Moreover, the decoded MIB provides the frame number and SSB block index, enabling the *Syncher* to determine slot numbers within each received subframe. This information is crucial for subsequent tasks like DCI search, PDSCH and PUSCH decoding as shown in Figure 2.

Once the cell search is complete, the *Syncher* transitions to the cell tracking stage to maintain synchronization with

the gNB. For subframes scheduled for SSB transmission, the *Syncher* continuously performs the correlation between the received signals and the predefined PSS/SSS sequences. This is to determine the offset of the currently received block and decode the MIB to update the current slot number, similar to cell search state, as described in the preceding paragraph. Using the offset, the *Syncher* also adjusts the number of samples to receive next time. This is to ensure that subsequently received blocks start precisely at the first OFDM symbol in the slot. This step is crucial, as it aligns the received block with the start of the slot, ensuring subsequent operations such as accurate Fast Fourier Transform (FFT) processing of OFDM symbols. This precise synchronization guarantees the subsequent downlink messages to be decoded correctly.

## 3.2 Extracting Cell information & UE Search

After completing the cell search, we synchronize with the gNB and proceed to decode the SIB1. This message provides the radio resource settings necessary to detect UEs connecting to the gNB and to decode subsequent communications (e.g., *Random Access Channel messages* and *RRC Setup Requests*, etc.). The *Broadcast Worker* component, as highlighted in Figure 2, performs two main tasks: *(i)* Locate and decode the SIB1 message, using the parameters from the MIB (as decoded by the *Syncher*), and *(ii)* after successfully decoding and applying the configuration from SIB1, the *Broadcast Worker* transitions to monitoring new UEs connecting to the gNB. Firstly, to decode the SIB1 message, the *Broadcast Worker* utilizes the `pdcch-ConfigSIB1` from the previously decoded MIB message. This is to determine the configurations for CORESET#0 and Search Space#0. Such parameters enable monitoring and detecting PDCCH/DCI scheduling information for SIB1. Then, once the DCI is decoded, key parameters e.g., resource block allocations are obtained and used to process the corresponding IQ samples and extract raw bytes, which in turn, are parsed as the SIB1 message. Failure in any of these steps (e.g., CRC error) leads to failure in decoding the SIB1 message.

Finally, after extracting the configuration information from SIB1 and applying it to the *Broadcast Worker* and pre-initialized *UETracker*, the *Broadcast Worker* transitions to searching for new UEs joining the network. To this end, the *Broadcast Worker* monitors downlink transmission of the `Random Access Response` (RAR) message, as this is the first message sent from the gNB in response to the UE that initiates the random access procedure. The *Broadcast Worker* extracts the Random Access Radio Network Temporary Identifier (RA-RNTI) parameter from the RAR message, which temporarily identifies UEs during the random access process. This RA-RNTI is used to kick-start the *UETracker* (See Figure 2) to track the newly detected UE, until Cell Radio Network Temporary Identifier (C-RNTI) is assigned to the UE after successful contention resolution.

To ensure continuous and real-time detection of new UEs connecting to the gNB, the *Broadcast Worker* persistently monitors each downlink transmission slot for the RAR message, which additionally carries the RAR uplink grant information. This grant enables the decoding of the subsequent uplink message (`RACH Message 3`), which might contain the `RRC Setup Request` from the UE. Therefore, the *Broadcast Worker* must detect and process the RAR message promptly. Once RAR is detected, *Broadcast Worker* activates a *UETracker* and quickly forwards the uplink grant to subsequent SNI5GECT components such that `RRC Setup Request` message can be captured. We note that the `RRC Setup Request` message includes the `Contention Resolution Identity`, a critical field required to carry out malformed `RRC Setup` message injection attacks [9]. For the UE to successfully pass the contention resolution process and accept our injected malformed `RRC Setup` message, it is necessary to extract the `Contention Resolution Identity` from the original `RRC Setup Request` message and include it in the malformed `RRC Setup` message. Thus, failure in decoding this message in real-time makes it impossible to launch attacks involving malformed `RRC Setup` messages [9].

## 3.3 Connection and Message Monitoring

This component of SNI5GECT (named "UETracker") monitors the communication between the gNB and the UE upon activation by the *Broadcast Worker*. It searches each slot to detect and decode messages between the gNB and the UE. Based on the status of the communication, it determines *when* to inject the message into the UE.

As shown in Figure 2, each instance of *UETracker* contains three main components: *(i)* ***UE DL Worker***, which decodes downlink messages from the gNB towards the UE; *(ii)* ***GNB UL Worker***, which decodes uplink messages from the UE to the gNB; and *(iii)* ***GNB DL Injector***, which encodes messages intended to be injected into the downlink communication from the gNB to the UE. The key functionalities performed by these components are outlined in Algorithm 1.

### 3.3.1 UE DL Worker: Decoding Messages from gNB

*UE DL Worker* processes and decodes the downlink signal from the gNB to the UE. This translates the time-domain signal into frequency components OFDM symbols. Moreover, since the attacker acts as a third party in the communication and not as the gNB, all scheduling information is obtained via passive over-the-air sniffing. To this end, immediately after being activated by the *Broadcast Worker*, the *UE DL Worker* utilizes *CORESET* and *Search Space* configurations obtained from SIB1 message to iteratively search and decode the PDCCH in each downlink slot (see lines 12-13 in Algorithm 1).

Next, the *UE DL Worker* applies these configurations from DCI to decode the PDSCH into raw bytes, which correspond

to messages transmitted from the gNB to the UE (Line 16 in Algorithm 1). Subsequently, if the received bytes pass the CRC check, we perform packet dissection and identify the protocol state on-the-fly (Lines 22-23 in Algorithm 1) via *WDissector* library [9]. We then either generate a targeted attack payload based on the protocol state or simply ignore the state (see procedure `process_decoded_message`) in Algorithm 1. We note that the attack payload generation is completely modular and arbitrarily extensible for testing. In our evaluation, we show such extensibility by evaluating existing attacks and by discovering new downgrade attacks.

Finally, the initially obtained *CORESET* and *Search Space* configuration contain values that are not applicable to the entire lifetime of the connection. Thus, the *UETracker* will miss those messages sent using the UE-specific configurations. Consequently, to ensure valid decoding of all user data messages, *UETracker* extracts UE-specific parameters (`CellGroupConfig`) from the *RRC_Setup* message and applies any relevant configuration changes to itself and associated components (see lines 18-20 in Algorithm 1).

This component is essential for decoding both uplink and downlink messages. On one hand, Downlink Control Information (DCI) for both uplink and downlink transmissions is sent in the downlink. Without accurately detecting and decoding this DCI, it is impossible to retrieve the scheduling information required to decode the actual PDSCH (downlink) and PUSCH (uplink) messages, thereby preventing successful retrieval of downlink and uplink data. Furthermore, while decoding the SIB1 message provides the *UETracker* with cell-wide configurations necessary for initial message detection and decoding, it does not include UE-specific configurations. Successfully detecting and decoding the *RRC_Setup* message is crucial for obtaining these UE-specific configurations, enabling the framework to fully track and decode messages exchanged between the gNB and the UE.

#### 3.3.2 GNB UL Worker: Decoding Messages from the UE

The *GNB UL Worker* decodes uplink messages sent from the UE to the gNB. However, processing of uplink signal is non-trivial. Firstly, 5G NR has two options for uplink waveforms (CP-OFDM and DFT-s-OFDM). Secondly, we note that in Frequency Division Duplex (FDD), the uplink frequency is different from the downlink frequency. More importantly, since the signals are transmitted over-the-air, the downlink and uplink signals are not perfectly aligned in the time domain. Hence, the demodulation of the uplink is performed independently of the downlink OFDM demodulation.

There are several ways to align with the timing of the UE transmission, such as making use of GPS Disciplined Oscillators (GPSDO) alongside the SDR. However, in our setup, we do not rely on GPSDO for SDR timing. Instead, the attacker synchronizes its clock with the gNB using the SSB blocks. Due to the hardware delay and transmission delay between the gNB and the attacker, some time misalignment still exists between them. Consequently, uplink signals are not correctly aligned from the attacker's perspective and directly decoding such misaligned subframes may result in decoding failure. Nonetheless, the gNB instructs the UE to adjust its data transmission timing using the Timing Advance Command (TAC). TAC contains the information about how early the UE has to send out the signal before the start of the subframe. This is to correctly align the signals from the UE with the start of the subframe when such signals arrive on the gNB side. Thus, when the UE transmits UL signals, an observer unaware of the TAC can only sniff the start of the downlink subframe, failing to decode UL signals that were transmitted before the downlink subframe. To avoid such constraint, SNI5GECT applies TAC correction to successfully decode the UL signals.

Since we use Time Division Duplex (TDD), both uplink and downlink share the same frequency. Consequently, uplink and downlink samples are received simultaneously. To correctly align the start of the uplink subframe with the start of the first OFDM symbol, some correction on the uplink symbols is required (see line 33 in Algorithm 1). Specifically, since the TAC instructs the UE to send out the signals before the start of the subframe, some uplink symbols for the current subframe reside in the last received block of samples. We utilize the TAC to compute the timing offset in the uplink subframe. This offset reduces the time misalignment between the attacker and the target UE, and ensures accurate decoding of most messages sent by the UE to the gNB, e.g., the critical messages exchanged before the security context is established. We evaluated the robustness of this approach in **RQ1**.

Since the *uplink grant* is sent via downlink from the gNB to the UE, the *GNB UL Worker* checks if there exists a grant for the current slot (see lines 27-30 in Algorithm 1). If such a grant exists, then the *GNB UL Worker* uses the scheduling information provided in the uplink grant, obtained from the DCI on the PDCCH, to decode the PUSCH. The *GNB UL Worker* then processes the assigned resource blocks to demodulate the received signal and extracts the raw bytes transmitted by the UE (see line 35 in Algorithm 1). Then, similar to *UE DL Worker*, if the decoded message passes the CRC check, it is sent to the *WDissector* library [9] for packet analysis and protocol state identification. This is then used for possible attack payload generation and injection (Lines 36-37).

#### 3.3.3 GNB DL Injector: Inject messages towards the UE

The *GNB DL Injector* is responsible for encoding and injection of messages intended to appear as if they originate from a legitimate gNB towards the tracked UE. Unlike the passive monitoring approach of the sniffing capability, the *GNB DL Injector* does not require scheduling information from the gNB. Instead, it generates its own scheduling information and injects it into the DCI DL, which is encoded into the PDCCH. Concurrently, the intended message to be injected towards the

**Algorithm 1** Message Processing from Subframe

```
Require: Subframes
 1: procedure PROCESS_DECODED_MESSAGE(message)
 2:     ▷ dissect the decoded message
 3:     packet ← WDWorker.dissect(message)
 4:     ▷ generate a targeted attack payload or ignore
 5:     packet_state ← WDWorker.detect_state(packet)
 6:     if is_attack_state(packet_state) then
 7:         payload ← generate_exploit(packet)
 8:         ▷ inject attack payload in downlink transmission
 9:         GNB_DL_Injector.inject(payload)
10:
11: for each subframe do
12:     for ⟨slot, slot_number⟩ ∈ subframe do
13:         ⟨dl_grant, ul_grant⟩ ← search_dci(slot, slot_number)
14:         if (dl_grant ≠ ∅) then
15:             ▷ UE DL Worker decodes the downlink message
16:             ⟨message, crc⟩ ← pdsch_decode(slot, dl_grant)
17:             ▷ apply configurations to SNI5GECT components
18:             if (checked(crc) ∧ message_type = RRC_Setup) then
19:                 configs ← extract_configs(message)
20:                 apply_configs(configs)
21:             ▷ generate state-aware attack payload
22:             if checked(crc) then
23:                 process_decoded_message(message)
24:         if (ul_grant ≠ ∅) then
25:             grant_queue.push(ul_grant)
26:         ▷ check the uplink grant from the head of grant queue
27:         UL_grant ← grant_queue.first
28:         ▷ slot number always increases sequentially
29:         ▷ hence, check whether UL_grant is for current slot
30:         if is_for_current_slot(UL_grant, slot_number) then
31:             ul_grant ← grant_queue.pop_head()
32:             ▷ apply offset to synchronize UE and attacker time
33:             apply_delay_calibration(slot)
34:             ▷ GNB UL Worker decodes the uplink message
35:             ⟨message, crc⟩ ← pusch_decode(slot, ul_grant)
36:             if checked(crc) then
37:                 process_decoded_message(message)
```

UE is encoded into the PDSCH using the time-domain and frequency-domain scheduling information from the DCI DL. When a response from the UE is required, the *GNB DL Injector* also generates a DCI to schedule an uplink transmission, encodes it into the PDCCH, and attaches it to the next slot. Eventually, all the encoded data is converted into IQ samples and transmitted to the UE (Line 9 in Algorithm 1).

It is possible that the injected message is transmitted in the same slot as the gNB's transmission, leading to collisions. If the signal strength is insufficient, this may result in jamming the UE's received signals. To mitigate this issue and improve the attack's success rate, we inject the same message multiple times in consecutive subframes. However, if we attempt to inject messages in both slots within the same subframe and continue over consecutive subframes, then the SDR fails to transmit as many samples. To address this, we inject the message only in the first slot of the subframe (see rightmost side of Figure 2). For the second slot, we either transmit a PDCCH carrying a DCI uplink grant, allowing us to schedule an uplink grant for the UE, or we copy 80% of the samples from the current slot and inject them into the second slot. This approach aims to block the gNB transmission and prevent the UE from accepting messages sent by the gNB. For most attacks, we only need to inject the messages into five consecutive subframes in order to successfully carry out the attack, which corresponds to 5ms in total.

However, injecting a message to the UE is a complicated process due to *(i)* inherent delays in transmitting and receiving signals to and from the SDR and the phone baseband hardware, and *(ii)* the fact that the UE's timing is synchronized with the gNB rather than the attacker (i.e., SNI5GECT acts as the third party in the communication). As a result, aligning the injection time with the UE's timing becomes complex, since a small difference in the injection time (range of microseconds) can result in a message arriving too early or too late. This, in turn, would cause the injected message to be dropped by the UE due to CRC errors. To overcome this challenge, SNI5GECT uses an iterative, offline search method to find the appropriate *delay calibration value* for message injection: This requires placing an SDR at potential target locations and running the IQ sample recorder. The process begins by initially synchronizing SNI5GECT with the gNB and then sending out an arbitrary message with a zero-delay offset. Then, on the recorder side, we first identify the slot that the message is injected. Finally, we use the decoding function from *UE DL Worker* to try to decode the injected signals with different delay offset, until the message can be successfully decoded. With this process, we aim to heuristically identify the timing offset required for successful decoding at the target UEs in the locations, resulting in successful injections. Hence, such *approximate* offset value, which resulted in successful decoding of injected signals, is used for subsequent message injection by SNI5GECT during attacks. We empirically show that to launch successful attacks, such offset does not need to change for a distance of at least 20$m$ (see **RQ2**).

SNI5GECT approach is feasible from the UE perspective. This is because in the absence of a message from gNB, the injected message arrives at the correct time, then the UE is unable to distinguish whether the message originates from the gNB or the attacker. Therefore, the UE decodes and accepts the message, and responds accordingly. Consequently, if the attack is launched via a fake gNB to crash the UE, then SNI5GECT can equally crash the UE by injecting the same payload without using the rogue gNB. Similarly, if the attack aims to retrieve information from the UE, then the UE will receive the uplink grant and respond to the attacker in the next slot as if the UE communicates to the gNB.

## 4 Evaluation Setup

**Hardware and Software Setup:** SNI5GECT is developed on top of srsRAN 4G [26] project, which offers both a 5G Standalone (SA) gNB prototype and an open-source 5G capable UE implementation. This project also provides functions required for DCI search, PDSCH decoding/encoding and PUSCH decoding. We wrote ≈10950 lines of C++ code (including header, test and attack modules) to integrate these functions together and implemented the components outlined in Section 2. We also leverage *WDissector* project [9] for real-time packet dissection and learning of protocol states.

We evaluated SNI5GECT using five commercial off-the-shelf (COTS) 5G SA-capable devices (see Table 1). Additionally, we evaluated SNI5GECT with two different legitimate 5G gNBs, with Open5GS [17] as the core network in both cases. For open-source RAN implementation, we use srsRAN with a USRP B210 as the radio front end for the legitimate 5G gNB. For commercial gNB, we employed a setup consisting of an Effnet gNB [1] paired with a Phluido Remote Radio Unit (RRU) [20] and a USRP B210 SDR as the legitimate gNB. SNI5GECT supports maximum 50MHz bandwidth in TDD and DCI formats 0/1_0/1, paired with USRP-B210 SDR. In our evaluation, both gNBs operated on a 20MHz n78 Band using TDD and a subcarrier spacing of 30 kHz. In total, our evaluation of SNI5GECT employs three USRP SDRs, one each for the legitimate 5G gNB and SNI5GECT framework (see Figure 1). A third USRP SDR is needed for the legitimate 4G eNB running srsRAN 4G, which, in turn, is used to detect downgrade attacks. The physical setup is shown in Figure 3.

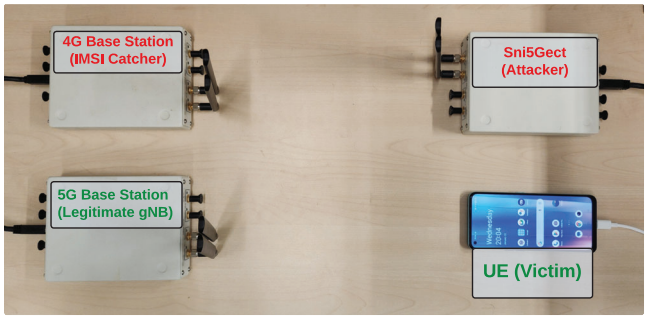


Figure 3: Physical setup showcasing a rogue 4G base station (IMSI catcher), a legitimate 5G base station, the SNI5GECT attacker device, and the victim UE (OnePlus Nord CE 2).

We conducted evaluations at two different distances: placing the phone close to the SNI5GECT SDR, marked as 0 meters, and positioning it 1 meter away from the SNI5GECT SDR, marked as 1 meter.

**Evaluation Protocol:** The evaluation process is conducted as follows. First, we start the legitimate 5G gNB and wait for UE to establish a connection. Next, we launch the SNI5GECT framework, allowing it to synchronize with the srsRAN gNB, once synchronized, the framework begins sniffing and detecting SIB1 messages. It then passively waits for the UE to establish the connection with the legitimate gNB. After this, we disable the airplane mode on the UE, causing it to connect to the gNB. After a few seconds, we turn on the airplane mode to disconnect the UE from the gNB and evaluate the results of the attack or sniffing process.

All our evaluations were conducted on a Debian 12 system, with the SNI5GECT software stack running inside an Ubuntu 22.04 Docker container. The host machine has an AMD Ryzen 9 5950X CPU and 32 GB DDR4 RAM.

Table 1: Tested 5G UEs, employed modem and patch version.

| Model | Modem | Patch Version |
|---|---|---|
| **OnePlus Nord CE 2 IV2201** | MediaTek MT6877V/ZA | 2023-05-05 |
| **Samsung Galaxy S22 SM-S901E/DS** | Snapdragon X65 | 2024-06-01 |
| **Google Pixel 7** | Exynos 5300 | 2023-05-05 |
| **Huawei P40 Pro ELS-NX9** | Balong 5000 | 2024-02-01 |
| **Fibocom FM150-AE USB modem** | Snapdragon X55 | NA |

Table 2: Effectiveness of SNI5GECT sniffing capability for different distance between the UE and the SNI5GECT SDR. Number of sniffed messages is provided alongside accuracy.

| Device | Dist. | Sniffing Accuracy | Uplink | Downlink |
|---|---|---|---|---|
| **OnePlus Nord CE 2** | 0 m | 93.32% (4418) | 75.77% (938) | 99.54% (3480) |
| | 1 m | 92.20% (8074) | 70.45% (1576) | 99.66% (6498) |
| **Huawei P40 Pro** | 0 m | 94.86% (1180) | 86.08% (371) | 99.51% (809) |
| | 1 m | 94.74% (1262) | 84.87% (387) | 99.89% (875) |
| **Pixel 7** | 0 m | 86.55% (1094) | 75.89% (491) | 97.73% (603) |
| | 1 m | 90.71% (1768) | 74.11% (435) | 97.87% (1333) |
| **Samsung Galaxy S22** | 0 m | 93.15% (1061) | 88.78% (467) | 96.90% (594) |
| | 1 m | 97.51% (745) | 94.68% (338) | 100% (407) |

## 5 Evaluation Results

### 5.1 RQ1: How effective is the sniffer?

We evaluate this question by first capturing the over-the-air communication between a smartphone and the gNB via the SNI5GECT sniffer and then comparing the total sniffed downlink and uplink messages exposed in the SNI5GECT trace file (i.e., PCAP) with the trace file exposed by the gNB itself (i.e., ground-truth). This allows us to calculate the effectiveness of SNI5GECT sniffer since the trace file generated by the gNB itself (via srsRAN) contains *all the messages* of the communication (i.e., downlink and uplink) and thus reveals whether the sniffer misses any messages within its trace file. The sniffer effectiveness is showcased in Table 2 over multiple connection attempts and with varying distances. Table 2 indicates that the sniffing effectiveness of the downlink channel is substantially better than the uplink channel, as respectively captured in columns *"Downlink Accuracy" and "Uplink Accuracy"*. Nonetheless, the overall accuracy of the sniffer is high, consistently capturing over 80% of over-the-air messages between the UE and the gNB as highlighted in "Accuracy" column.

The reduced uplink sniffing accuracy is attributed to failure in detecting the DCI for uplink messages, which results in

SNI5GECT missing several uplink messages. Additionally, SNI5GECT works as a third party in the communication, and hence, the timing of the uplink does not perfectly align with the downlink timing from the gNB. Despite SNI5GECT making use of TAC to compensate the uplink timing difference and leveraging the cyclic prefix of the OFDM symbols to tolerate some timing errors, decoding of the PUSCH channel occasionally fails, resulting in missed uplink messages. These factors collectively explain the reduced accuracy.

We also evaluated the robustness of our uplink sniffing approach. In particular, SNI5GECT leverages the initial Timing Advance (TA) Command value provided by the gNB (i.e., sniffed from RAR message). This is to reduce the time misalignment difference between the attacker and the target UE for uplink sniffing (see Section 3.3.2). We designed two different experiments to evaluate the effectiveness of this approach. For the first experiment, we evaluate the uplink sniffing performance for UEs far from the SNI5GECT. Firstly, we fixed the position of SNI5GECT and the gNB, with the gNB six meters away from SNI5GECT. Then, we place the four phones five meters away from the SNI5GECT, toggle the airplane mode to trigger the connection, and finally compare the uplink sniffing result with the ground truth from the srsRAN base station. We gradually increase the distance with a step of five meters until the phones are 20 meters away from SNI5GECT. For each distance, we run the evaluation ten times. Results are shown in Figure 4. We observe that the uplink sniffing performance remains generally high (often $70\% - 95\%$) within 20m distance and the performances generally decrease (up to 55%) as the distance between SNI5GECT and the target UE increases.

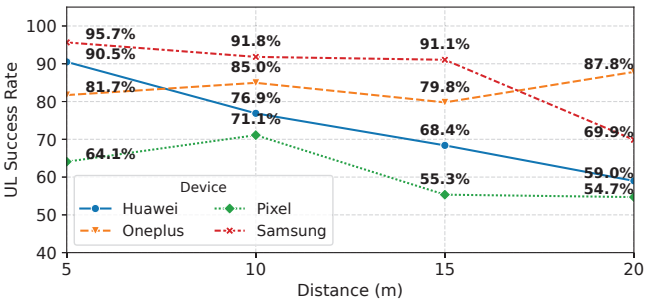


Figure 4: Success Rate of Uplink Sniffing w.r.t distance

The previous experiment is affected by the signal quality, as the antenna used might receive poor signals from the UE when it is far away. We observed that the RSRP drops from 21.55 dBm to 8.95 dBm on average from 5 meters to 20 meters away. Thus, to evaluate the robustness of uplink sniffing, we also designed another experiment. In this evaluation, we placed the phones one meter away from both the SNI5GECT and the gNB to get better signal quality. Four phones connect to the gNB simultaneously and we record the IQ samples sent between the gNB and the target UE into a file. Then we run the uplink sniffing test offline by directly reading the signals from the pre-recorded files. This guarantees each test have exactly the same set of input. Moreover, in addition to applying the TA from RAR directly, we also evaluate the uplink sniffing performance within the range [TA-10, TA+10] with a step of two. For each TA offset, we evaluate with ten different connection recording files. This helps us evaluate the robustness of adopting the TA Command from the gNB to the UE, without being affected by other factors such as signal qualities. We illustrate the success rate for different UEs in Figure 5. Indeed, we observe that the uplink sniffing performance remains similar within the window of TA$\pm$4. This robustness zone corresponds to a timing misalignment of $\pm 1.04$ microseconds, $\approx \pm 312$ meters. This is acceptable for effective uplink decoding under realistic conditions.

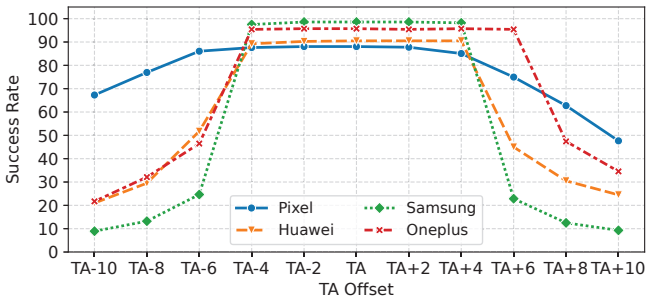


Figure 5: Uplink Sniffing w.r.t TA command offset

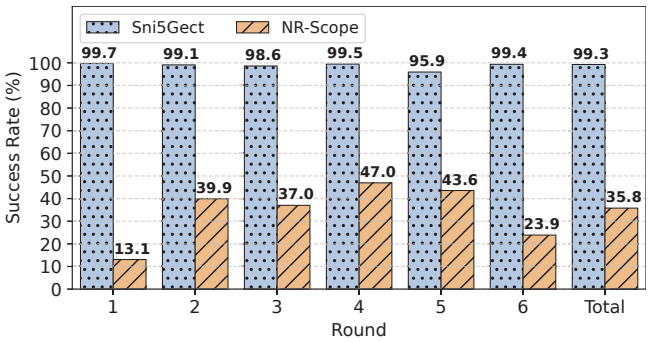


Figure 6: Success Rate of DCI search

Finally, we also compare the accuracy of our DCI search with respect to existing framework i.e., NR-Scope [31], using the same setup described in Figure 3. In this evaluation, we connect four different UEs to the target srsRAN gNB simultaneously. Concurrently, SNI5GECT passively sniffs the DCIs transmitted from the gNB to the UEs. We then compare the number of captured DCIs with the ground truth obtained from the srsRAN base station logs. We run the same evaluation for six rounds, and evaluated the success rate of each round. Figure 6 illustrates the competitive performance of SNI5GECT over NR-Scope [31]. During our experiment, NR-Scope often failed to detect new UE connections and hangs frequently. As a result, it missed a lot of DCIs, thus leading to its poor perfor-

mance. We observe that in contrast to NR-Scope, SNI5GECT effectively captures new UEs approaching to the gNB, keeps track of their UE specific configurations, and then utilize these configurations to search and decode the DCIs sent from the gNB to the target UE.

## 5.2 RQ2: How effective is stateful injection?

We evaluate the effectiveness of SNI5GECT in successfully injecting arbitrary messages at any given state during the 5G communication. We firstly inject messages in different states (e.g., after `RRC Setup Request`) and then confirm whether such messages are successfully received by the UE. To this end, we use Fibocom USB modem as target UE and extract its internal capture file (ground-truth) to confirm the reception of an injected message into the modem firmware. Additionally, we evaluate the success rate with respect to the number of injected messages. This is particularly insightful to reveal whether injecting multiple times the same message can significantly increase the chances for a message to be received by the UE, in case of over-the-air failures or collisions.

Table 3 showcases the total and successfully injected messages and success rate. For each row, Table 3 also contains the results given how many times (*"Duplication"* column) the same message was injected for each state. Overall, SNI5GECT achieves a high success rate ($> 80\%$) of message injection in most evaluated states. Furthermore, increasing the number of *Duplication* reveals a significant increase in the success rate of messages injected after `Authentication Failure` and after `Security Mode Complete`. This is expected due to the increased chance of collisions with many uplink messages during such states and during injection.

We also evaluated the effectiveness of SNI5GECT to attack the victim UE at different distances. This is to demonstrate the effectiveness and robustness of the injection time offset found by brute force. Similar to previous experiments, we placed the UE five to 20 meters away from the SNI5GECT (while keeping the distance between SNI5GECT and the gNB six meters) and a 30dB amplifier is placed at the TX port of the SNI5GECT SDR to increase the injection signal power. Then, we inject the `Registration Reject` [15] after `Registration Request` state. We run this attack in three different rounds and in each round, we toggle the connection for about 20 times. Finally, we check the `pcap file` collected from QCSuper [23] to confirm the message is actually received at the expected state on the UE side. We observe over 95% success rate within 20m and the success rate decreased to about 83% at about 20m distance (see Table 4). This demonstrates SNI5GECT's robustness in realistic condition i.e., to inject messages into communication even when the victim UE is relatively far. It also demonstrated that with increased signal power, the injection success also increases.

Overall, our results reveal that SNI5GECT is not only a practical offensive tool, but it also tests the resiliency of currently deployed 5G networks against over-the-air intrusions.

## 5.3 RQ3: Evaluation of 5G Attacks

At a high level, we evaluate SNI5GECT with three different attack categiories with the following requirements (Table 5): *(i)* ***One-Shot Attacks*** require injecting a single message towards the UE. If the UE accepts the message, it reacts immediately (e.g., by downgrading its network connection or crashing); *(ii)* ***One-Shot Attacks with Response***, which in addition to sending a single message to the UE, the attack also expects a corresponding response from the UE. *(iii)* ***Multi-Stage Attacks*** depend on injecting the same or different messages multiple times at different states to successfully execute the attack. Consequently, SNI5GECT must perform multiple injections and/or message sniffing to achieve the desired outcome. We also evaluate attacks triggering *crash*, *downgrade*, *fingerprinting* and other vulnerabilities from prior research e.g., 5Ghoul [9] and *5GBaseChecker* [30].

### 5.3.1 One-Shot Attacks

**5Ghoul Attacks:** 5Ghoul [9] highlights several malformed messages that can lead UE modems to crash. To show the practicality of SNI5GECT, we selected five representative One-Shot attacks (out of 14) to replicate via our framework: four instances of the malformed `RRC Setup` attack and one instance of the malformed Radio Link Control (RLC) attack, all of which are reported to crash the UE modem. Then, we evaluated these attacks using the *OnePlus Nord CE2* smartphone, which employs a vulnerable MediaTek modem. The success rate of each attack across different distances is shown in Table 6 (see column *srsRAN gNB*). While few attacks resulted in performance around 70%, a change of the attacker's distance consistently results in a higher accuracy of above 90%. Attack failure occasionally happens when the UE does not accept messages from SNI5GECT. This is due to the approximated

injection delay, which may not yield the optimal delay for all devices (see Section 3.3.3). Nonetheless, SNI5GECT *jams* downlink messages from gNB and resends its previously injected message to increase the attack success rate. Once the jamming stops, the UE proceeds to communicate with the gNB. Finally, we confirm 5G modem crashes via ADB logs, which include error strings e.g., `"sModemReason"`.

**Registration Reject Downgrade Attack:** We evaluate SNI5GECT to show its effectiveness in performing *inter-generation downgrade attacks* [15]. Such an attack requires only a single message to be injected into the communication (`Registration Reject - N1 mode not allowed`) without further interaction with the UE. Concretely, the expected behavior of a successful attack includes *(i)* the UE switching from 5G to 4G connectivity and *(ii)* the 5G gNB persistently attempting to recover the connection by repeatedly sending messages such as `Security Mode Command`

Table 3: SNI5GECT Injection Performance.

| Duplication # | RRC Setup Request | | | Registration Request | | | Authentication Failure | | | Security Mode Complete | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | # Messages | # Injected | Success Rate | # Messages | # Injected | Success Rate | # Messages | # Injected | Success Rate | # Messages | # Injected | Success Rate |
| **1** | 356 | 306 | 85.96% | 232 | 181 | 78.02% | 47 | 29 | 61.70% | 51 | 23 | 45.10% |
| **2** | 251 | 219 | 87.25% | 423 | 337 | 79.67% | 72 | 60 | 83.33% | 67 | 60 | 89.55% |
| **3** | 316 | 272 | 86.08% | 350 | 262 | 74.86% | 42 | 28 | 66.67% | 46 | 44 | 95.65% |
| **4** | 234 | 202 | 86.32% | 364 | 297 | 81.59% | 71 | 51 | 71.83% | 43 | 39 | 90.70% |

Table 4: Injection Performance at Different Distances.

| Distance | Total | Success | Success Rate |
|---|---|---|---|
| 5m | 73 | 70 | 95.89% |
| 10m | 65 | 61 | 93.85% |
| 15m | 73 | 69 | 94.52% |
| 20m | 73 | 61 | 83.56% |

Table 5: Attacks Category and Expected Outcomes.

| Attacks | Category | Outcomes |
|---|---|---|
| **5Ghoul Attacks** | One-shot Attacks | Crash |
| **Registration Reject Injection** | One-shot Attacks | Downgrade |
| **Identity Request Injection** | One-shot Attack expecting Response | Fingerprinting |
| **Authentication Replay Injection** | Multi-stage Attacks | Downgrade |
| **5G AKA Bypass** | Multi-stage Attacks | Security Bypassing |

towards the UE. Hence, we use our physical setup with both 5G and 4G base stations to evaluate the attack success rate (see Figure 3). The attack behavior was repeatedly observed during our tests (see attack success rate in the leftmost side of Figure 7), thus confirming the effectiveness of SNI5GECT to reliably reproduce downgrade attacks. We note that downgrade attack may also occasionally fail due to the brute-force delay calibration approach mentioned earlier.

### 5.3.2 One-Shot Attacks with Response

The 5G standard enhances security by introducing the Subscription Permanent Identifier (SUPI) as a permanent identifier to replace the less secure IMSI from 4G. Upon receiving an `Identity Request` from the gNB, the UE encrypts the SUPI to generate the Subscription Concealed Identifier (SUCI) and only transmits the SUCI back to the network. This encryption prevents attackers from directly deriving the SUPI from the SUCI. However, as demonstrated in the 5G SUCI-Catchers [3] paper, attackers can still track UE devices with the encrypted SUPI, thereby compromising user privacy.

Table 6: 5Ghoul attacks under srsRAN or Effnet gNBs.

| CVE | Dist. | srsRAN gNB | | Effnet gNB | |
|---|---|---|---|---|---|
| | | Total | Success rate | Total | Success rate |
| CVE-2023-20702 | 0 m | 18 | 100.00% | 26 | 92.31% |
| | 1 m | 59 | 100.00% | 28 | 85.71% |
| CVE-2023-32843 | 0 m | 19 | 100.00% | 25 | 68.00% |
| | 1 m | 22 | 90.91% | 22 | 90.91% |
| CVE-2023-32842 | 0 m | 22 | 77.27% | 25 | 76.00% |
| | 1 m | 21 | 95.24% | 24 | 75.00% |
| CVE-2024-20003 | 0 m | 22 | 90.91% | 32 | 40.62% |
| | 1 m | 22 | 100.00% | 23 | 56.52% |
| CVE-2023-32845 | 0 m | 15 | 86.67% | 25 | 64.00% |
| | 1 m | 20 | 95.00% | 20 | 95.00% |

To showcase the capability of SNI5GECT to perform such attacks, we inject an `Identity Request` message into the target UE and consider the attack successful only if the UE responds with the expected *Identity Response* message, which contains the UE's SUCI. To evaluate the success rate of this attack, we first start a legitimate 5G gNB and then launch the attack against various phones and different distances. The results are presented in the center of Figure 7. We observe that most phones achieve a success rate of approximately 60% with some variability. As mentioned earlier, the static injection delay causes the UE to occasionally ignore the injected `Identity Request` messages. Furthermore, since this scenario relies on receiving a response from the UE, a failure by the sniffer to successfully decode the message results in missing the `Identity Response`, leading to attack failures.

### 5.3.3 Multi-Stage Attacks

**Novel 5G Downgrade:** We have discovered a *novel downgrade attack* and its risk is acknowledged by GSMA [10].

Our attack relies on forcibly starting timer T3520 within the UE by injecting a replayed *Authentication Request* (Auth. Req.) message containing an invalid sequence number (SQN). According to the 3GPP specification, once the UE receives such replayed message, the UE replies with an Authentication Failure message, starts timer T3520 and blacklists (i.e., mark as barred) the currently connected 5G gNB if the authentication procedure is not completed before expiry of such timer or the authentication procedure keeps failing. Once the UE blacklists the gNB, it disconnects from 5G and connects to a nearby 4G eNB with the same MCC and MNC as the previously connected gNB instead. Furthermore, if no 4G eNBs are available, the UE does not attempt connecting to the same gNB even after waiting a long time.

Moreover, to prevent the gNB from retrying the authentication procedure, SNI5GECT injects the replayed *Authentication Request* message immediately after the *Registration Request* message and continues to do so after receiving any *Authentication Failure* message from the UE. This forces the UE to drop the connection and blacklist the gNB regardless of subsequent attempts from the gNB to continue with the authentication procedure. This demonstrates SNI5GECT's Multi-Stage attack capability, as it injects messages at different stages of the connection process while dynamically updating the message content to reflect changing conditions.

To evaluate this attack, we start a legitimate gNB and test

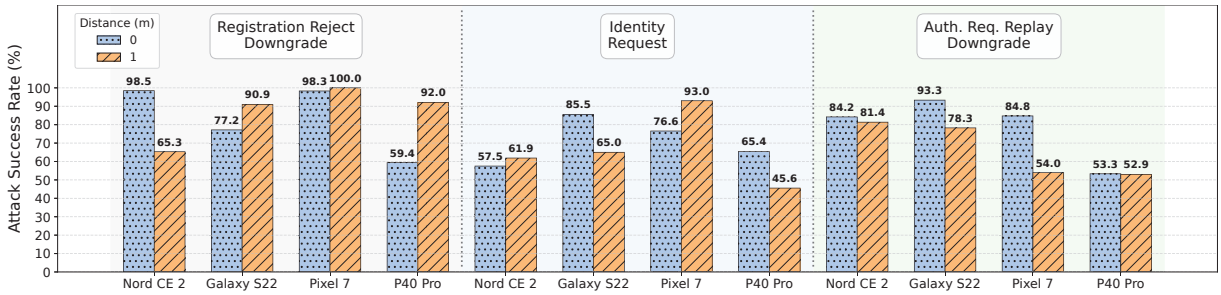


Figure 7: Attack Success Rates by Device, Distance, and Attack Type using **srsRAN** as legitimate gNB.

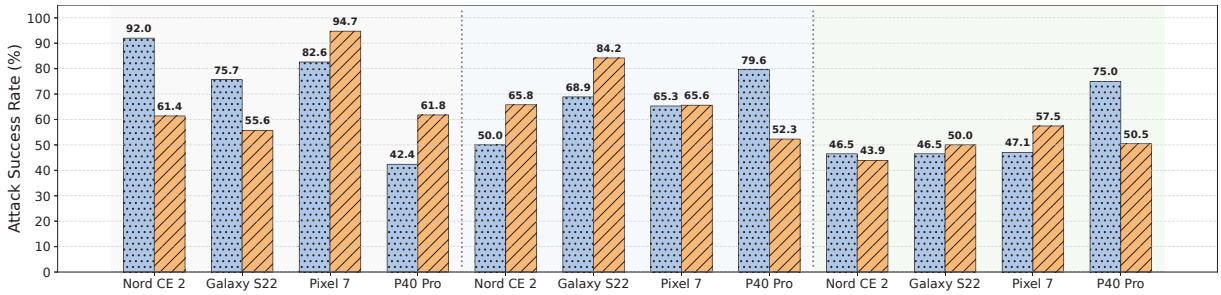


Figure 8: Attack Success Rates by Device, Distance, and Attack Type using **Effnet** as legitimate gNB.

various phones against the attack. The attack is deemed successful if the UE replies with multiple *Authentication Failure* messages, disconnects and does not connect back to the corresponding gNB. We observe that *Pixel 7* always recovers immediately and reconnects back to the gNB after attack. In contrast, other phones downgrade to 4G and remain in the downgraded state for several minutes to hours. The summary of results is presented in the rightmost side of Figure 7.

Overall, the downgrade is highly successful against most of the evaluated phones ($>$ 80%), except for *Pixel 7* at 1*m* distance and *Huawei P40*, both of which result in a moderate success rate of 50%. Such reduction on the success rate for some phones can be attributed to failure of SNI5GECT to capture every (*Authentication Failure*) response from the UE, allowing the gNB to retry the authentication procedure and thus nullify the attack attempt. Nonetheless, such results still highlight the practicality of the novel downgrade attack.

We also evaluated that SNI5GECT can successfully launch such complex attack even if the victim UE is 20 meters away. This further demonstrates the practicalities of using SNI5GECT framework to test and carry out complex attacks. In summary, these results highlight that SNI5GECT not only can be used to reproduce 5G attacks in practical settings, but also show its potential to discover new attacks on 5G UEs.

**5GBaseChecker AKA Bypass:** We also evaluate security bypass vulnerabilities e.g., *5G AKA Bypass* from 5GBaseChecker [30]. For this experiment, we use the reported vulnerable *Pixel 7* smartphone. We received expected plaintext response `Registration Complete` and `PDU Session Establishment Request` from the UE, as stated in the original attack, immediately after injecting plaintext

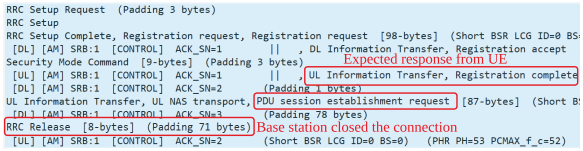


Figure 9: Capture of plaintext Registration Accept message.

`Registration Accept` message. This also shows that the message injection by SNI5GECT was successful. Concurrently, the gNB had sent a `Security Mode Command`, but it was ignored by the UE due to the injected `Registration Accept` message. However, as `Registration Complete` was not the expected UE response, the core network reports an error and then the gNB sends the `RRC Release` message to terminate the connection (see Figure 9). This experiment also shows that reported 5G attacks may not have the same impact when exploited in the wild.

### 5.3.4 Implications of 5G Downgrade Attacks

In this experiment, we show the implications of downgrading a UE connection to use a lower network generation by launching the IMSI-catching attack [24] via a rogue 4G eNB.

Unlike 5G, in which the UE permanent identity is informed through the concealed SUCI, 4G allows the UE to inform its International Mobile Subscriber Identity (IMSI) in plaintext. This 4G vulnerability is well known and makes it possible for attackers to extract the UE SUCI to track the UE location over time, probe the UE for targeted attacks and perform further privacy breaches. To carry out this attack, an attacker sets up

| Work | Scope | | Sniffing | | | Downlink Overshadow | | | Generation | |
|---|---|---|---|---|---|---|---|---|---|---|
| | pre-AKA | post-AKA | PDCCH | PDSCH | PUSCH | Synch. | Broad. | Uni. | LTE | 5G |
| NR-Scope [31] | ● | ❍ | ● | ● | ❍ | ❍ | ❍ | ❍ | ❍ | ● |
| LTESniffer [12] | ● | ● | ● | ● | ● | ❍ | ❍ | ❍ | ● | ❍ |
| Falcon [8] | ● | ● | ● | ❍ | ❍ | ❍ | ❍ | ❍ | ● | ❍ |
| SigOver [32] | - | - | - | - | - | ● | ● | ❍ | ● | ❍ |
| AdaptOver [6] | ● | ❍ | ● | ● | ● | ◗ | ◗ | ● | ● | ❍ |
| SigUnder [18] | - | - | - | - | - | ● | ● | ❍ | ❍ | ● |
| 5GSniffer [19] | ❍ | ● | ● | ❍ | ❍ | ❍ | ❍ | ❍ | ❍ | ● |
| SNI5GECT | ● | ❍ | ● | ● | ● | ◗ | ◗ | ● | ❍ | ● |

Table 7: SNI5GECT with respect to similar works. ●: supported, ❍: unsupported, ◗: Not evaluated, - : Not applicable. Synch.: Synchronization, Broad.: Broadcast, Uni.: Unicast.

a fake 4G eNB and tricks a downgraded UE to connect to it. Once connected, the attacker sends an `Identity Request` message to the UE, which reveals its IMSI by replying to the 4G eNB with a plaintext `Identity Response` message.

We use the *OnePlus Nord CE2* phone to demonstrate this attack. We firstly launch the attack as described in Section 5.3.3 to forcibly downgrade the UE to 4G. At the same time, we run a modified srsRAN 4G eNB as an IMSI-Catcher to extract the UE IMSI once the UE downgrades. Since the phone is configured to prioritize 5G NR over 4G, the UE first approaches the 5G gNB. When the SNI5GECT attacker detects the new UE approaching, it sends an `Authentication Request` message to the UE and the UE responds with `Authentication Failure`. After injecting the message three times, the UE disconnects from the 5G gNB, connects to the IMSI-Catcher rogue eNB and responds with the `Identity Response` message, thus indicating a successful attack.

### 5.3.5 Experiments with commercial gNB

To assess SNI5GECT's attack capability in real-world scenarios, we evaluate against UEs connecting to a commercial gNB, employing *Effnet's Radio Access Network (RAN)* and a Phluido Remote Radio Unit (RRU) software [20]. As the commercial gNB does not provide the ground-truth capture file, we skip evaluation of the sniffing accuracy and evaluate concrete attacks such as 5Ghoul, *Registration Reject*, *Authentication Replay* and *Identity Request*.

The results are highlighted in Table 6 (see column `Effnet`) and Figure 8. Overall, the attack success rate for UEs communicating with Effnet gNB is slightly lower as compared to srsRAN. This is because the commercial gNB transmits messages at a faster rate, which reduces the availability of empty slots and hence, increases the chance of collisions during SNI5GECT message injection. Additionally, Effnet gNB might send multiple messages before the UE moves to the next state, which prevents messages injected too early from being accepted by the UE. SNI5GECT mitigates such issues by injecting a message multiple times (cf. Section 3.3.3).

## 6 Related Work

Table 7 positions SNI5GECT with respect to similar frameworks. In the following, we discuss this in detail.

**5G Sniffing:** 5GSniffer [19] allows sniffing of certain messages (e.g., *MIB, DCI*). This facilitates traffic analysis and indirectly tracking the UE by exploiting privacy leaks during communication. However, in contrast to SNI5GECT, 5GSniffer does not handle the majority of 5G physical layer channels and hence cannot sniff any message payload during stateful procedures e.g., *RRC Attach* and *NAS Registration*. Therefore, SNI5GECT is not only useful in privacy-related attacks, but also in stateful spoofing attacks that require extracting parameters from UE messages (e.g., downgrade in Section 5.3.3).

Recent work such as NR-Scope [31] is incapable to sniff the uplink, which is fundamental for attacks occurring after specific UE responses (statefulness). QCSuper [23], MobileInsight [28] and *Network Signal Guru* [29] can extract over-the-air messages directly from the smartphone by reverse engineering of diagnostic protocol. However, they cannot sniff messages of other UEs, as sniffing only reveals messages of its own communication with the gNB. Consequently, such tools cannot be used to attack arbitrary UEs. Additionally, they rely on proprietary non real-time protocol, hence, launching stateful attacks is not applicable when compared to SNI5GECT.

**Spoofing Attacks in 5G:** SigOver [32] and Sigunder [18] introduce over-the-air spoofing of broadcast and paging messages in 4G and 5G networks to execute DoS attacks via overshadowing techniques. In comparison, SNI5GECT focuses on more complex (stateful) procedures such as *RRC Attach* and *NAS Registration*, thus enabling access to a broader attack surface. AdaptOver [6] enables injection of arbitrary messages toward the UE in mobile networks. However, AdaptOver is tailored to 4G and only focuses on static "One-Shot" attacks that do not react to UE replies and depicts high latency during downlink injection (e.g., 50*ms*). In contrast, SNI5GECT targets 5G and can inject messages based on UE responses within only 5*ms*. Additionally, AdaptOver is not publicly available while SNI5GECT is open-source (See availability).

**Downgrade Attacks in Mobile Networks:** Downgrade attacks studied in prior works [2, 15, 21] are mostly evaluated from the perspective of a rogue gNB. In comparison, SNI5GECT focuses on the practical side of launching a variety of attacks, including but not limited to downgrades. As an example, we find a novel downgrade attack (see Section 5.3.3) as a byproduct of our framework. Therefore, we argue that SNI5GECT complements such prior works by serving as a practical evaluation tool as opposed to competing.

**5G Security Testing and Verification:** Several prior works support over-the-air testing and exploitation of UEs [2, 9, 16, 30] using a rogue gNB, which, needs to be configured with the UE credentials. SNI5GECT avoids limitations of knowing UE credentials by acting as a third-party in the communication, and thus, by freely injecting messages at pre-authentication states. Other work like 5GReplay [22] is not applicable in over-the-air scenarios and target only RRC and NAS protocol. In contrast, SNI5GECT offers additional control on lower pro-

tocol layers e.g., RLC/PDCP/MAC. Finally, orthogonal to our work, model-based verification focuses on finding 5G vulnerabilities through sole analysis of 3GPP specification [13, 14]. Nonetheless, SNI5GECT complements such works to assess the realistic impact (see AKA Bypass in Section 5.3.3).

**Detection of Overshadowing Attacks:** Physical-layer attacks can be detected and mitigated by relying on detecting a rogue gNB [4, 25] or overshadowing and jamming of static information e.g., the SIB [5, 11]. However, these works do not address selective overshadowing of 5G messages. Moreover, the 5G sniffer in SNI5GECT enables future research on passive physical-layer or protocol-level 5G attack detection.

## 7 Threats of Validity and Limitations

**Generalization of framework:** SNI5GECT only supports 5G and downlink injection. However, its modular design can facilitate extensions to support earlier mobile technologies e.g., 4G. Additionally, future work will focus on optimizing DCI search and message coding to enable uplink injection.

**TAC Handling:** As the distance between UE and attacker grows, our use of the TA is less applicable to the attacker, thus affecting uplink sniffing accuracy. However, we show that SNI5GECT is not affected by TA variations up to a delta of $\pm 4$ as shown in Figure 5.

**Overshadowing Synchronization:** Our static delay offset for downlink injection remains unchanged for up to 20m (Table 4) with attack accuracy $\approx 80\%$. However, our overshadowing synchronization approach requires significant manual effort to build a timing offset map based on the attacker-UE distance. Even with this map, SNI5GECT cannot precisely determine the UE locations. Thus, synchronizing with UEs within the entire coverage area remains challenging, as different delay offsets must be tried until success. Therefore, building a practical 5G sniffer targeting both pre-RAR and post-RAR states and signal overshadowing framework with unknown UE distance, remains an open research direction.

**Attacking Post-Authentication States:** SNI5GECT relies in sniffing plain-text messages before injecting attacks via overshadowing. Consequently, SNI5GECT is currently not applicable to exploit post-authentication procedures due to encryption of messages, which leads the UE to ignore unencrypted payloads by design. However, we plan to expand SNI5GECT to target plain-text control messages (MAC) that are accepted even after post-authentication states.

**Sniffing Post-RAR states:** Currently, SNI5GECT does not sniff UEs already connected to the gNB (post-rar states) since it requires tracking the UE's RNTI from the start of the PRACH procedure (see Section 2.2). We note that decoding post-rar messages in 5G is still a fundamental challenge, as it requires brute forcing the DCI, which is time consuming and impractical for real-time attacks in the pre-authentication states focused by SNI5GECT. Nonetheless, future works can mitigate such real-time barrier by integrating SNI5GECT with a brute-force DCI search algorithm offloaded to GPUs.

**Sensitivity to Physical Layer Behaviour:** Currently, SNI5GECT is used indoors or within a maximum $\approx 20m$ distance from the UE. Therefore, physical layer behaviour associated with long distance and dense environments (e.g., multipath propagation, beamforming, adaptive power control) is not currently considered. We intend to build a foundational tool for experimenting with stateful 5G message sniffing and injection. However, we plan to expand SNI5GECT evaluation to account for physical layer behaviours that commonly occur in 5G networks deployed in urban environments.

**Physical Layer Support:** SNI5GECT supports Single Input Single Output (SISO) in in 5G TDD configuration. Multiple Input Multiple Output (MIMO) and other orthogonal physical layer features can be introduced to SNI5GECT by migrating code from OpenAirInterface [7] and srsRAN project [27].

**Identifying and distinguishing UEs:** The random RNTI uniquely assigned to each UE during the PRACH process is leveraged by SNI5GECT to sniff and attack such UE without affecting UEs associated with different RNTIs. However, since 5G is designed to avoid user tracking, SNI5GECT cannot distinguish a smartphone model or user (i.e., victim UE) solely based on the RNTI to launch targeted attacks. Nonetheless, we consider such fingerprinting an orthogonal area to our work and envision future research in such area to complement the scope of SNI5GECT, rather than competing with it.

## 8 Conclusion

This paper showcases SNI5GECT– an open-source framework that performs over-the-air packet sniffing and injection in 5G NR networks without using a rogue gNB. Consequently, this allows security researchers to quickly perform security testing and evaluate the realistic impact of 5G attacks against arbitrary UEs. This advantage is evident by SNI5GECT capability to launch challenging, multi-stage attacks that require real-time injection and sniffing of messages in different stateful 5G procedures. Such attacks are exemplified by our evaluation of *AKA Bypass* and our newly discovered downgrade attack (see Section 5). Furthermore, our evaluation of SNI5GECT against five COTS UEs reveals a high accuracy in sniffing ($> 80\%$) and injection ($> 70\%$) of messages. Therefore, we argue that SNI5GECT is a fundamental tool in 5G security research that enables not only over-the-air 5G exploitation but advancing future research on packet-level 5G intrusion detection and mitigation, security enhancements to 5G physical layer security and beyond. To the best of our knowledge, there is no open-source alternative that offers the capabilities of SNI5GECT, which is publicly available in the following URL: https://github.com/asset-group/Sni5Gect-5GNR-sniffing-and-exploitation

## Acknowledgement

We thank the anonymous shepherd and reviewers for their insightful comments on our paper. This research is partially supported by National Research Foundation, Singapore, under its National Satellite of Excellence Programme “Design Science and Technology for Secure Critical Infrastructure: Phase II" (Award No: NRF-NCR25-NSOE05-0001) and MOE Tier 2 grant (Award number MOE-T2EP20122-0015). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the respective funding agencies.

## Research Ethics

Our work, which falls on the wireless security area, often involves taking part in responsible security disclosure. This is important to not only ensure that affected parties are aware of our work, but to also to ensure that appropriate fixes or counter measures are put in place. Therefore, during development and experimentation with SNI5GECT, we made sure to contact the affected special interest group (GSMA) as to inform them of the contents of our research.

During our development and evaluation, in order to minimize the impact to other people around, we setup the test bed using test country code MCC and test network code MNC and make sure only the devices under test are been sniffed and attacked.

Additionally, we only plan to publicly release the SNI5GECT framework and the exploits discussed in the paper. Other serious exploits leveraging the framework will not be publicly available to avoid abusing SNI5GECT to launch attacks against people’s smartphone. Instead, to foster security research, we will separately make the exploits available via requests to a publicly form. This is so we can verify that we are sending the exploits to trusted institutions like universities and research institutions. After all, our goal is to heavily foster security research on mobile networks in a responsible manner.

## Open Science Commitment

We have released our tool open source with this publication, including all of the components mentioned in Section 3. The Zenodo link (https://doi.org/10.5281/zenodo.15601773) provides the current implementation of all of these components and instructions on how to run SNI5GECT and replicate the results of our evaluation (Section 5). We also make the exploits available in the repository. This guarantees researchers can run and reproduce experiments using our framework and run further evaluations.

## References

[1] Effnet AB. Effnet 5G solutions. https://www.effnet.com/products/protocolstack-nw/, 2025. Accessed: 2025-01-08.

[2] Evangelos Bitsikas, Syed Khandker, Ahmad Salous, Aanjhan Ranganathan, Roger Piqueras Jover, and Christina Pöpper. Ue security reloaded: Developing a 5g standalone user-side security testing framework. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, page 121–132, New York, NY, USA, 2023. Association for Computing Machinery.

[3] Merlin Chlosta, David Rupprecht, Christina Pöpper, and Thorsten Holz. 5g suci-catchers: still catching them all? In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec ’21, page 359–364, New York, NY, USA, 2021. Association for Computing Machinery.

[4] Adrian Dabrowski, Nicola Pianta, Thomas Klepp, Martin Mulazzani, and Edgar Weippl. Imsi-catch me if you can: Imsi-catcher-catchers. In *Proceedings of the 30th Annual Computer Security Applications Conference*, ACSAC ’14, page 246–255, New York, NY, USA, 2014. Association for Computing Machinery.

[5] Jiongyu Dai, Usama Saeed, Ying Wang, Yanjun Pan, Haining Wang, Kevin T. Kornegay, and Lingjia Liu. Detection of overshadowing attack in 4g and 5g networks. *IEEE/ACM Trans. Netw.*, 32(6):4615–4628, October 2024.

[6] Simon Erni, Martin Kotuliak, Patrick Leu, Marc Roeschlin, and Srdjan Capkun. Adaptover: adaptive overshadowing attacks in cellular networks. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, MobiCom ’22, page 743–755, New York, NY, USA, 2022. Association for Computing Machinery.

[7] Eurecom. OpenAirInterface 5g wireless implementation. https://gitlab.eurecom.fr/oai/openairinterface5g, 2025. Accessed: 2025-01-08.

[8] Robert Falkenberg and Christian Wietfeld. FALCON: an accurate real-time monitor for client-based mobile network data analytics. In *2019 IEEE Global Communications Conference, GLOBECOM 2019, Waikoloa, HI, USA, December 9-13, 2019*, pages 1–7. IEEE, 2019.

[9] Matheus E. Garbelini, Zewen Shang, Shijie Luo, Sudipta Chattopadhyay, Sumei Sun, and Ernest Kurniawan. 5Ghoul: Unleashing chaos on 5g edge devices. https://asset-group.github.io/disclosures/5ghoul/, 2023. Accessed: 2025-01-07.

[10] GSMA. Coordinated vulnerability disclosure (CVD) programme. https://www.gsma.com/solutions-and-impact/technologies/security/gsma-coordinated-vulnerability-disclosure-programme/, 2025. Accessed: 2025-01-18.

[11] Virgil Hamici-Aubert, Julien Saint-Martin, Renzo E. Navas, Georgios Z. Papadopoulos, Guillaume Doyen, and Xavier Lagrange. Leveraging overshadowing for time-delay attacks in 4g/5g cellular networks: An empirical assessment. In *Proceedings of the 19th International Conference on Availability, Reliability and Security*, ARES '24, New York, NY, USA, 2024. Association for Computing Machinery.

[12] Tuan Dinh Hoang, CheolJun Park, Mincheol Son, Taekkyung Oh, Sangwook Bae, Junho Ahn, Beomseok Oh, and Yongdae Kim. Ltesniffer: An open-source LTE downlink/uplink eavesdropper. In Ioana Boureanu, Steve Schneider, Bradley Reaves, and Nils Ole Tippenhauer, editors, *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2023, Guildford, United Kingdom, 29 May 2023 - 1 June 2023*, pages 43–48. ACM, 2023.

[13] Syed Rafiul Hussain, Mitziu Echeverria, Imtiaz Karim, Omar Chowdhury, and Elisa Bertino. 5greasoner: A property-directed security and privacy analysis framework for 5g cellular network protocol. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, page 669–684, New York, NY, USA, 2019. Association for Computing Machinery.

[14] Abdullah Al Ishtiaq, Sarkar Snigdha Sarathi Das, Syed Md Mukit Rashid, Ali Ranjbar, Kai Tu, Tianwei Wu, Zhezheng Song, Weixuan Wang, Mujtahid Akon, Rui Zhang, and Syed Rafiul Hussain. Hermes: Unlocking security analysis of cellular network protocols by synthesizing finite state machines from natural language specifications. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 4445–4462, Philadelphia, PA, August 2024. USENIX Association.

[15] Bedran Karakoc, Nils Fürste, David Rupprecht, and Katharina Kohls. Never let me down again: Bidding-down attacks and mitigations in 5g and 4g. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, page 97–108, New York, NY, USA, 2023. Association for Computing Machinery.

[16] Syed Khandker, Michele Guerra, Evangelos Bitsikas, Roger Piqueras Jover, Aanjhan Ranganathan, and Christina Pöpper. Astra-5g: Automated over-the-air security testing and research architecture for 5g sa devices. In *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '24, page 89–100, New York, NY, USA, 2024. Association for Computing Machinery.

[17] Sukchan Lee. Open5GS a open source implementation for 5G core and EPC. https://open5gs.org/open5gs/, 2025. Accessed: 2025-01-08.

[18] Norbert Ludant and Guevara Noubir. Sigunder: a stealthy 5g low power attack and defenses. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '21, page 250–260, New York, NY, USA, 2021. Association for Computing Machinery.

[19] Norbert Ludant, Pieter Robyns, and Guevara Noubir. From 5g sniffing to harvesting leakages of privacy-preserving messengers. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 3146–3161, 2023.

[20] Phluido. Interoperable and portable, software-defined open RAN physical layer. https://www.phluido.net/, 2025. Accessed: 2025-01-18.

[21] David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. Breaking lte on layer two. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1121–1136, 2019.

[22] Zujany Salazar, Huu Nghia Nguyen, Wissam Mallouli, Ana R. Cavalli, and Edgardo Montes de Oca. 5greplay: a 5G network traffic fuzzer - application to attack injection. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*. Association for Computing Machinery, 2021.

[23] P1 Security. Qcsuper: A tool for capturing raw 2g/3g/4g radio frames. https://github.com/P1sec/QCSuper, 2025. Accessed: 2025-01-08.

[24] Altaf Shaik, Jean-Pierre Seifert, Ravishankar Borgaonkar, N. Asokan, and Valtteri Niemi. Practical attacks against privacy and availability in 4g/lte mobile communication systems. In *NDSS*. The Internet Society, 2016.

[25] Jisoo Shin, Yongyoon Shin, and Jong-Geun Park. Network detection of fake base station using automatic neighbour relation in self-organizing networks. In *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, pages 968–970, 2022.

[26] Software Radio Systems. Open source sdr 4g software suite from software radio systems (srs). https://github.com/srsran/srsRAN_4G, 2025. Accessed: 2025-01-08.

[27] Software Radio Systems. srsRAN: open source O-RAN 5G CU/DU solution from software radio systems (srs). https://github.com/srsran/srsRAN_Project, 2025. Accessed: 2025-01-08.

[28] MobileInsight team. Mobileinsight. http://www.mobileinsight.net/, 2025. Accessed: 2025-01-08.

[29] QTRUN Technologies. Network signal guru (nsg). https://play.google.com/store/apps/details?id=com.qtrun.QuickTest, 2025. Accessed: 2025-01-08.

[30] Kai Tu, Abdullah Al Ishtiaq, Syed Md Mukit Rashid, Yilu Dong, Weixuan Wang, Tianwei Wu, and Syed Rafiul Hussain. Logic gone astray: A security analysis framework for the control plane protocols of 5g basebands. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 3063–3080, Philadelphia, PA, August 2024. USENIX Association.

[31] Haoran Wan, Xuyang Cao, Alexander Marder, and Kyle Jamieson. NR-Scope: A practical 5g standalone telemetry tool. In *Proceedings of the 20th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '24, page 73–80, New York, NY, USA, 2024. Association for Computing Machinery.

[32] Hojoon Yang, Sangwook Bae, Mincheol Son, Hongil Kim, Song Min Kim, and Yongdae Kim. Hiding in plain signal: Physical signal overshadowing attack on LTE. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 55–72, Santa Clara, CA, August 2019. USENIX Association.

## Appendix

Table 8: Evaluation of Registration Reject Downgrade Attack using srsRAN as legitimate gNB.

| Device | Distance(m) | Total | Success | Success Rate |
|---|---|---|---|---|
| **OnePlus Nord CE 2** | 0 | 196 | 193 | 98.47% |
| | 1 | 248 | 162 | 65.32% |
| **Samsung Galaxy S22** | 0 | 127 | 98 | 77.17% |
| | 1 | 99 | 90 | 90.91% |
| **Pixel 7** | 0 | 59 | 58 | 98.31% |
| | 1 | 198 | 198 | 100% |
| **Huawei P40 Pro** | 0 | 96 | 57 | 59.38% |
| | 1 | 200 | 184 | 92.00% |

Table 9: Evaluation of Identity Request using srsRAN as legitimate gNB.

| Device | Distance(m) | Total | Success | Success Rate |
|---|---|---|---|---|
| **OnePlus Nord CE 2** | 0 | 113 | 65 | 57.52% |
| | 1 | 160 | 99 | 61.88% |
| **Samsung Galaxy S22** | 0 | 159 | 136 | 85.53% |
| | 1 | 300 | 195 | 65.00% |
| **Pixel 7** | 0 | 64 | 49 | 76.56% |
| | 1 | 100 | 93 | 93.00% |
| **Huawei P40 Pro** | 0 | 107 | 70 | 65.42% |
| | 1 | 90 | 41 | 45.56% |

Table 10: *Auth. Req. Replay Downgrade* under srsRAN gNB.

| Device | Distance(m) | Total | Success | Success Rate |
|---|---|---|---|---|
| **OnePlus Nord CE 2** | 0 | 57 | 48 | 84.21% |
| | 1 | 59 | 48 | 81.36% |
| **Samsung Galaxy S22** | 0 | 45 | 42 | 93.33% |
| | 1 | 46 | 36 | 78.26% |
| **Pixel 7** | 0 | 164 | 139 | 84.76% |
| | 1 | 63 | 34 | 53.97% |
| **Huawei P40 Pro** | 0 | 45 | 24 | 53.33% |
| | 1 | 68 | 36 | 52.94% |

Table 11: *Registration Reject Downgrade* under Effnet gNB.

| Device | Distance(m) | Total | Success | Success Rate |
|---|---|---|---|---|
| **OnePlus Nord CE 2** | 0 | 25 | 23 | 92.00% |
| | 1 | 57 | 35 | 61.40% |
| **Samsung Galaxy S22** | 0 | 111 | 84 | 75.68% |
| | 1 | 115 | 64 | 55.65% |
| **Pixel 7** | 0 | 46 | 38 | 82.61% |
| | 1 | 38 | 36 | 94.74% |
| **Huawei P40 Pro** | 0 | 92 | 39 | 42.39% |
| | 1 | 157 | 97 | 61.78% |

Table 12: *Identity Request Attack* under Effnet gNB.

| Device | **Distance(m)** | **Total** | **Success** | **Success Rate** |
|---|---|---|---|---|
| **OnePlus Nord CE 2** | 0 | 38 | 19 | 50.00% |
| | 1 | 38 | 25 | 65.79% |
| **Samsung Galaxy S22** | 0 | 45 | 31 | 68.89% |
| | 1 | 38 | 32 | 84.21% |
| **Pixel 7** | 0 | 49 | 32 | 65.31% |
| | 1 | 32 | 21 | 65.63% |
| **Huawei P40 Pro** | 0 | 49 | 39 | 79.59% |
| | 1 | 65 | 34 | 52.31% |

Table 13: *Auth. Req. Replay Downgrade* under Effnet gNB.

| **Device** | **Distance(m)** | **Total** | **Success** | **Success Rate** |
|---|---|---|---|---|
| **OnePlus Nord CE 2** | 0 | 43 | 20 | 46.51% |
| | 1 | 41 | 18 | 43.90% |
| **Samsung Galaxy S22** | 0 | 43 | 20 | 46.51% |
| | 1 | 30 | 15 | 50.00% |
| **Pixel 7** | 0 | 34 | 16 | 47.06% |
| | 1 | 47 | 27 | 57.45% |
| **Huawei P40 Pro** | 0 | 48 | 36 | 75.00% |
| | 1 | 109 | 55 | 50.46% |

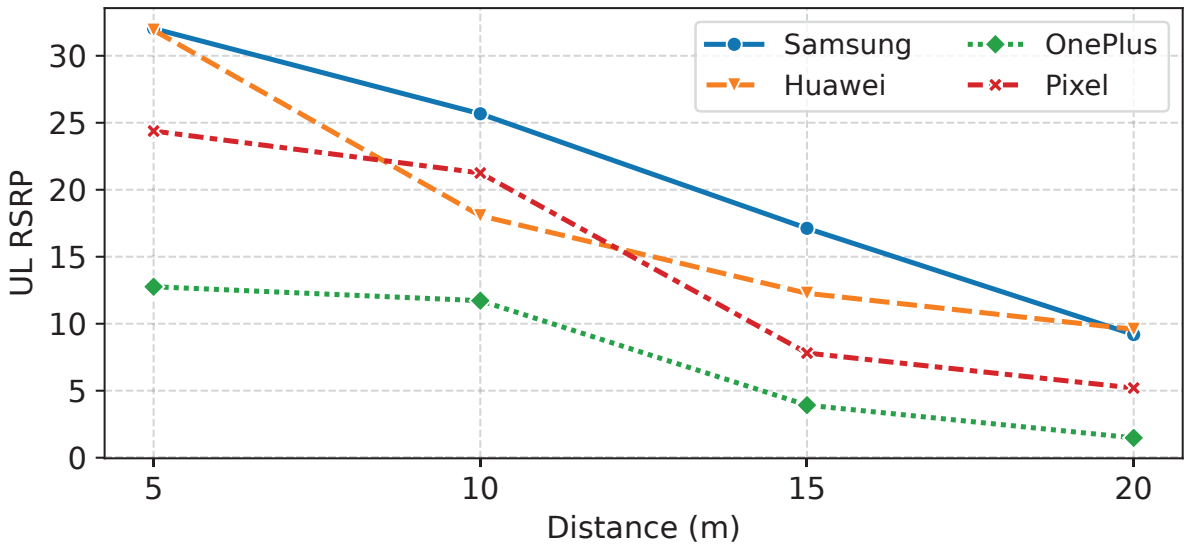


Figure 10: RSRP of Uplink Sniffing w.r.t distance

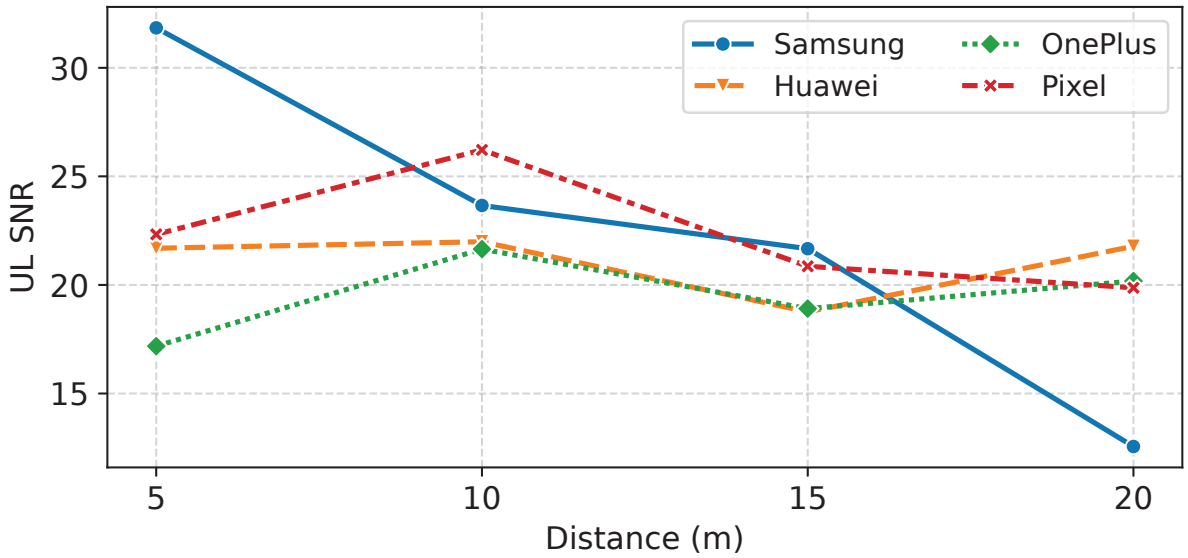


Figure 11: SNR of Uplink Sniffing w.r.t distance