Paging Storm Attacks against 4G/LTE Networks from Regional Android Botnets: Rationale, Practicality, and Implications

Kaiming Fang
Department of Computer Science
Binghamton University, State University of New York
kfang2@binghamton.edu

Guanhua Yan
Department of Computer Science
Binghamton University, State University of New York
ghyan@binghamton.edu

ABSTRACT

Although the impact of mobile botnet attacks against cellular networks has been studied in a number of previous works, little attention has been paid to regional botnets, where bot-infected mobile devices are geographically concentrated at local areas. In this work we investigate a new type of threats called *paging storm attacks*, which can be launched from a regional botnet to exhaust the limited paging capacity of cells in a 4G/LTE (Long-Term Evolution) network. As paging storm attacks can delay paging requests for legitimate time-critical voice or video calls in a target area, their real-life implications include user annoyance, distortion of call center analytics, and loss of productivity. To demonstrate the feasibility of such attacks, we design and implement a proof-of-concept Android botnet that can coordinate bot activities to create pulsating paging requests within a short period of time. We mathematically analyze the probability that normal paging requests are delayed due to a botnet attack. Experimental results observed from a high-fidelity emulation testbed reveal that paging storm attacks launched from a regional botnet can create repetitive surges of paging requests in the target LTE network, thereby delaying time-critical voice/video calls by several seconds.

CCS CONCEPTS

• **Security and privacy → Mobile and wireless security**.

KEYWORDS

Paging storm attacks, 4G/LTE networks, mobile botnets, Android

ACM Reference Format:
Kaiming Fang and Guanhua Yan. 2020. Paging Storm Attacks against 4G/LTE Networks from Regional Android Botnets: Rationale, Practicality, and Implications. In *13th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '20), July 8–10, 2020, Linz (Virtual Event), Austria.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3395351.3399347

1 INTRODUCTION

The importance of 4G/LTE (Long-Term Evolution) network infrastructure to society calls for thorough investigation of their resilience against mobile botnet attacks. Although the impact of mobile botnet attacks against cellular networks has been studied in a number of previous works [14, 18, 34, 49, 50], little attention has been paid to regional botnets, which recruit bots by infecting mobile devices geographically concentrated at local areas. Such regional botnets can be constructed by distributing bot malware as localized mobile apps, which have been gaining popularity in the mobile industry due to their ability in increasing return-on-investment [15]. Using regional botnets, it is possible to attack the cellular network infrastructure shared by their infected mobile devices, even though the mobile malware penetration rate can still be low world wide [35].

In this work we investigate a new type of threats called *paging storm attacks*, which can be launched from a regional botnet. The intuition behind such attacks is simple: if the attacker can use a regional botnet to exhaust the limited paging capacity of a base station, legitimate paging requests have to be delayed in the target area. Albeit conceptually easy to understand, paging storm attacks are not straightforward to carry out practically. First, to exhaust the paging capacity of a cell, bot activities in the regional botnet have to be highly coordinated so that the paging requests triggered can be pulsated within a very short period of time. However, as revealed in a few recent works [30, 38, 39, 44], it is difficult to create coordinated and decentralized pulsating attacks in distributed systems. Second, a paging request is triggered inside the LTE network only if the receiving terminal is in an idle state. It is not easy for a bot malware to infer whether its residing mobile phone is in an idle state, as such internal states are only available in the cellular modem; it is even more difficult to find another mobile device that is idle so sending a message to it would trigger a new paging request inside the network. Last but not least, even if a bot malware manages to infer its residing mobile device to be idle, how to report such a state to the botmaster server becomes a paradox because doing so through the LTE network would break the idleness of the device!

Against this backdrop, we propose a novel mobile botnet, in which each bot infers the idleness of its residing mobile device due to lack of cellular data transmissions. Bot malware on active devices regularly send beacon messages to the botmaster server, while those on idle ones do not. If the number of idle devices inferred exceeds a certain threshold, the botmaster commands active devices to immediately send short messages to the idle ones. A surge of paging requests generated due to these messages within a short period of time creates a pulsating paging storm attack in the network. As the bot-infected mobile devices in a regional botnet are likely to stay in the same area, it is possible that the pulsating paging requests can overload the paging capacity of target base stations, thus delaying time-critical services such as voice and video calls.

Existing efforts on mobile botnet attacks are mostly focused on design and analysis of *conceptual mobile botnets* [17, 33, 36, 45, 53]

WiSec '20, July 8–10, 2020, Linz (Virtual Event), Austria

ACM ISBN 978-1-4503-8006-5/20/07…$15.00
https://doi.org/10.1145/3395351.3399347

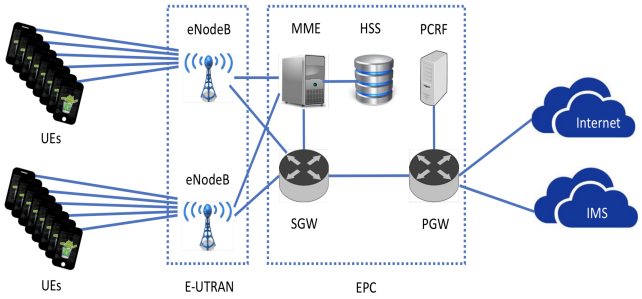


Figure 1: LTE network architecture

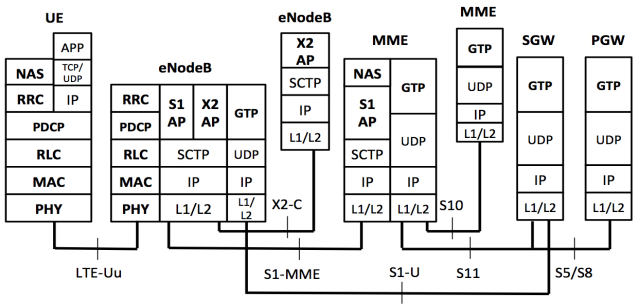


Figure 2: LTE network protocol stacks

and their attack effects are usually assessed through back-of-the-envelope calculations or coarse-grained simulations. In contrast to these previous efforts, our work validates the practicality of paging storm attacks by implementing a proof-of-concept Android botnet with bot malware needing only a few permissions from infected Android devices. Moreover, we build a high-fidelity emulation testbed to evaluate the realistic consequences of paging storm attacks from the proof-of-concept Android botnet.

Our main contributions are summarized as follows: (1) We identify the key hurdles involved in carrying out paging storm attacks, such as inferring the idleness of each malware-infected mobile device in order to create paging requests and the paradox that having an idle device report its inactivity through the LTE network interferes with its current network state. We then design and implement a proof-of-concept Android botnet that overcomes these challenges. (2) We mathematically analyze the probability that a legitimate paging request can be delayed by a paging storm attack and present numerical results that shed light on how the effects of paging storm attacks are affected by the botnet size and key LTE network parameters. (3) To study the resilience of 4G/LTE networks against paging storm attacks, we develop a high-fidelity emulation testbed that connects a state-of-the-art LTE network emulator and the official Android Emulator within which Android applications can be dynamically executed. We implement some missing features needed to study the effects of paging storm attacks truthfully. (4) Using this testbed, we perform a variety of experiments to gain insights into the resilience of an LTE network against paging storm attacks carried out by the proof-of-concept Android botnet. Our experimental results reveal that paging storm attacks from a regional botnet can delay time-critical voice/video calls by several seconds, for which the real-life implications include user annoyances, distortion of call center analytics, and loss of productivity.

2 PRIMER ON 4G/LTE NETWORKS

In order to explain how paging storm attacks work in 4G/LTE networks, we first introduce their paging procedures. Figure 1 illustrates the architecture of a typical 4G/LTE network, with its protocol stacks shown in Figure 2. To use LTE network services, a UE (User Equipment), such as a mobile phone, connects with a RAN (Remote Access Network) called E-UTRAN (Evolved Universal Terrestrial Radio Access Network), which consists of a number of base stations called *eNodeBs*. An eNodeB is connected to the all-IP core of an LTE network, which is called EPC (Evolved Packet Core). In the control plane, an *MME (Mobility Management Entity)* serves as the main signaling node as it keeps track of the location of each UE at the tracking area level, mutually authenticates each UE with aid of the *HSS (Home Subscriber Server)*, which is a database that contains authentication and billing information of the subscribers, and selects appropriate gateways for user traffic sessions in the user plane. In the user plane, an LTE network is connected to an external PDN (Packet Data Network), such as the public Internet or an IMS (IP Multimedia Subsystem), through a *PGW (PDN Gateway)*. The PGW interacts with the *PCRF (Policy and Charging Rules Function) server* for the network service policy, the QoS (Quality of Service) setting information for each user session, and the charging rule for each user account. The *SGW (Serving Gateway)* serves as the anchor point that forwards user traffic between eNodeBs and PGWs.

UE attachment to the network. When a UE attaches itself to the network, it is authenticated and registered within the network, resources such as EPS (Evolved Packet System) bearers are allocated for its data sessions, and mobility management functions are activated to keep track of its location inside the network. Henceforth, the UE's states are managed by two separate protocol layers: the RRC (Radio Resource Control) layer of the serving eNbodeB and the NAS (Non-Access Stratum) layer of the serving MME (see Figure 2). Moreover, the NAS layer of the MME keeps multiple sets of states for the UE, including EMM (EPS Mobility Management) states and ECM (EPS Connection Management) states. A newly attached UE is in a joint state of `RRC-CONNECTED` (an RRC connection has been established between the UE and the eNodeB), `EMM-REGISTERED` (the UE is attached, an IP address has been assigned to it, an EPS bearer has been established, and the MME knows its location at the tracking area level), and `ECM-CONNECTED` (an NAS signaling connection is established between the UE and the MME and radio and network resources have been allocated for the UE).

S1 release due to user inactivity. As seen in Figure 2, the S1 interface is used between an eNodeB and the EPC. On detection of the UE being idle for a while, the S1 release procedure can be triggered by the eNodeB to destroy the S1 bearer between the eNodeB and the SGW in the user plane as well as the S1 signaling connection between the eNodeB and the MME in the control plane. After S1 release, the UE is in a joint state of `RRC-IDLE` (no RRC connection has been established between the UE and the eNodeB), `EMM-REGISTERED`, and `ECM-IDLE` (no NAS signaling connection is established between the UE and the MME and no physical resources are allocated). Hence, for an idle but still attached UE, the MME

Table 1: Parameters needed to calculate paging frames and paging occasions

Notation	Meaning	Values	Source
SFN	System Frame Number	{0, 1, ..., 1023}	
T_u	UE-specific DRX cycle	{32, 64, 128, 256} (in frames)	MME
T_c	Cell-specific DRX cycle	{32, 64, 128, 256} (in frames)	eNodeB
T	$\min\{T_u, T_c\}$	$\min\{T_u, T_c\}$ (in frames)	
B	Total number of paging occasions in a DRX cycle	{4T, 2T, T, T/2, T/4, T/8, T/16, T/32 }	eNodeB
$UEID$	UE Identity Index Value	UE's IMSI mod 1024	UE
N	Number of paging frames per DRX cycle	$\min\{T, B\}$	
N_s	Number of subframes used for paging within a paging frame	$\max\{1, B/T\}$	
Ind	Index pointing to a predefined table for calculating paging occasion	$\lfloor UEID/N \rfloor \mod N_s$	
R	Maximum number of UE identities on a paging record list	[1, ..., 16]	eNodeB

knows its location at the tracking area level, but its RRC and NAS signaling connections are removed from the network and all the radio and network resources allocated to it are released.

Paging. The network notifies an idle but still attached UE of various events (e.g., an incoming call) using a *paging* process. As the MME keeps track of each UE in an `EMM-REGISTERED` state at the tracking area level, a paging message is sent by the MME to all the eNodeBs within the current tracking areas of the UE and each eNodeB notified delivers the paging request to the intended UE through its downlink physical channels. In LTE, both uplink and downlink transmissions are done in consecutive radio frames (*frames* for short), which are numbered repeatedly from 0 to 1023 by their *SFNs (System Frame Numbers)*. Each frame contains 10 *subframes*, the duration of which is one millisecond. Symbols within a subframe are organized into various physical channels. The eNodeB decides which UEs should be given the resources to send or receive data through a scheduling process at a subframe level.

An idle but still attached UE has its device radio operating at low power and listens to the control traffic on the PDCCH (Physical Downlink Control Channel) from the eNodeB to check if there are any incoming *paging* requests. If an idle UE monitors the PDCCH for paging requests in every subframe, its battery power would be drained quickly. To overcome this issue, a UE in an `RRC-IDLE` state that uses *DRX (Discontinuous Reception)* for paging wakes up only once every DRX cycle, which includes between 32 and 256 frames, or equivalently between 0.32 and 2.56 seconds.

To describe how a UE decides which frames and subframes it should monitor for paging requests in a DRX cycle, a few notations are introduced in Table 1. A frame is treated by both the UE and the eNodeB as a *paging frame* if the following holds for its SFN:

$$SFN \mod T = (T/N) \times (UEID \mod N) \quad (1)$$

For instance, given $T_c = 64$, $T_u = 128$, $B = T/4$, and $UEID = 20$, we have $T = 64$, $B = 16$, and $N = 16$, so the paging frames include those whose SFNs modulo 64 are equal to 16.

The subframe within a paging frame containing the UE's paging information is called its *paging occasion*. In the same example, we have $N_s = 1$ and $Ind = 0$ where definitions of N_s and Ind are given in Table 1. Multiple UEs may share the same paging occasion. At its paging occasions, a UE wakes up and monitors the PDCCH channel for P-RNTI (Paging Radio Network Temporary Identifier), which is a paging indicator of fixed value 0xFFFE. If the UE detects P-RNTI, it continues to demodulate and decode the RRC paging message transmitted on the PDSCH (Physical Downlink Shared Channel). Each RRC paging message contains a paging record list of at most R identities, where the maximum value of R is 16. If the UE finds its own identity on the paging record list, it initiates a service request to the MME; otherwise, it goes back to a sleeping mode and waits for its paging occasion in the next DRX cycle.

3 RATIONALE OF PAGING STORM ATTACKS

One key observation about the paging procedure of a 4G/LTE network is that the number of paging requests deliverable within a DRX cycle is at most $B \cdot R$, where we recall B is the total number of paging occasions in a DRX cycle and R the maximum number of identities on the paging record list in an RRC paging message. Hence, *if an attacker uses a regional botnet to create pulsating paging requests delivered to the same base station*, they may exhaust all the R slots for a paging occasion so legitimate paging requests mapped to the same paging occasion have to be delayed until the next DRX cycle. Similarly, these legitimate paging requests can be further delayed if the next paging occasion is still overwhelmed by the attack paging requests or other legitimate ones.

By delaying time-critical voice/video calls, real-life implications of paging storm attacks include the following. *(1) User annoyances:* A caller may feel frustrated if her call cannot get through quickly and abandon her call prematurely. *(2) Distortion of call center analytics:* The industry standard for the voice channels of a call center is to provide sufficient staffing so that 80% of voice calls be answered within 20 seconds [5]. Due to delayed call setup by paging storm attacks, some customers may give up their calls before getting connected to the call center or before their calls are answered. These situations distort call center analytics and cause miscalculation of staffing level. *(3) Loss of productivity and even human life:* Mission-critical service providers (e.g., police, fire brigade, ambulance, and first responders) rely on 4G/LTE networks for instant communications [21]. Delayed calls may not only lead to loss of productivity by these services but also loss of human life in emergency situations (e.g., heart attacks, earthquakes, fire, and mass shooting).

Mathematical analysis. We derive the likelihood that normal paging requests can be delayed because of a paging storm attack, using the notations in Table 1. As there are in total B paging occasions, each supporting page requests destined to at most R distinct UEs, a legitimate paging request may be delayed if its paging occasion

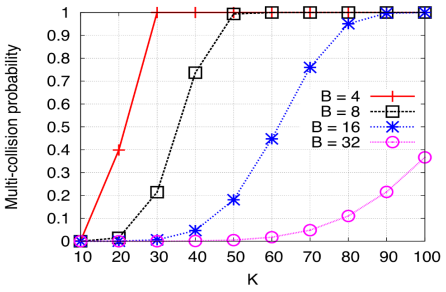


Figure 3: Analysis of paging storm attack effects ($K = K_l + K_i$)

is shared by more than R paging requests waiting to be delivered within the same DRX cycle. We consider an ideal scenario where all the paging requests destined to the idle bots arrive at the eNodeB within a short period of time that is less than a DRX cycle, assuming that the variation of transmission delays of both short messages and paging requests is negligible due to both their small sizes and use of high-speed wired transmissions in the core network. Suppose that at the time of the botnet attack, there are K_l legitimate paging requests and K_i paging requests destined to idle bots within the cell of interest. We also assume that within the cell the paging occasion for each paging request, regardless of its being legitimate or not, is *uniformly* distributed over all the B choices in the same DRX cycle.

We analyze *the increased probability of having a paging occasion with more than R paging requests waiting to be delivered due to the paging storm attack*. The problem can be modeled as an extension of the famous birthday problem, which is called *the birthday paradox for multi-collisions* [47]. The multi-collision problem asks: given that there are n buckets and q identical balls are independently thrown into these buckets, what is the probability that there is at least one bucket with no fewer than s balls? The birthday problem is a special case of the multi-collision problem with $s = 2$. Let $C(n, q, s)$ denote the solution to the multi-collision problem. According to [47], $C(n, q, s)$ can be calculated recursively:

$$C(n,q,s) = \frac{1}{n^{s-1}} \sum_{i=s}^{q} \binom{i-1}{s-1} (1-\frac{1}{n})^{i-s} (1 - C(n-1, i-s, s)),$$

where $C(n, q, s) = 0$ for $q < s$ and $C(n, q, s) = 1$ for $n = 1$ and $q \geq s$.

When multi-collisions occur, paging requests that fall into their corresponding paging occasions have to wait for the next DRX cycle. Hence, the effects of a paging storm attack can be modeled as the multi-collision probability under $n = B$, $q = K_l + K_i$, and $s = R + 1$. In Figure 3, we show the multi-collision probability, $C(B, K = K_l + K_i, R + 1)$ with $B \in \{4, 8, 16\}$, $K \in \{10, 20, ..., 100\}$, and $R = 7$ (which is recommended for LTE 5MHz bandwidth). For example, when $B = 8$ and $K_l = 30$, adding 10 more paging requests from the botnet increases the multi-collision probability by 2.4 times from 0.214 to 0.737; when $B = 16$ and $K_l = 50$, adding 20 more paging requests from the botnet increases the multi-collision probability by 3.2 times from 0.181 to 0.760. Hence, a paging storm attack can increase the probability of delayed paging requests significantly.

4 PROOF-OF-CONCEPT ANDROID BOTNET

In this section, we present the design and implementation of a regional botnet for paging storm attacks.

4.1 High-level botnet design

The bot malware mimics a legitimate mobile application targeting a local area. A few high-level design issues are discussed as follows.

First, bot activities are coordinated through a centralized botmaster server. We do not assume that individual bots should know the existence of other peers unless the botmaster informs them. Moreover, we do not consider peer-to-peer mobile botnets because the high transmission delays of C&C (Command and Control) messages in such overlay networks make it difficult to create pulsating paging storm attacks. The botmaster can be deployed on any publicly accessible server. C&C messages transmitted between bots and the botmaster can be encrypted to make their detection difficult.

Second, attack paging requests are triggered through SMS messages. We choose SMS because it is ubiquitously deployed and is more stealthy than phone or video calls. Although not perfect, SMS is still a desirable choice here. Late delivery of short messages can occur due to a variety of reasons [10]. For example, the receiving terminal may not be ready or experience weak signals at the time of delivery, the observable delivery receipts arrive much later than the real SMS contents due to their low priority in transmission, or texting during heavy network use can also affect text delivery speed. In many of these cases, sending SMS messages can still trigger immediate paging requests to the receiving terminals, although the messages may be eventually delivered to these devices later.

Last, the botnet is designed to be resilient against easy mitigation. As the botnet attack is aimed at exhausting the paging capacity of a target cell, a straightforward defense mechanism seems to be reconfiguring the network parameters to increase a cell's paging capacity. Idle-mode paging configurations are contained in SIB2 (System Information Block 2) in LTE. A mitigation scheme that modifies paging configurations to increase a cell's paging capacity requires paging messages to inform UEs in `RRC-IDLE` or `RRC-CONNECTED` about system information changes [7]. Such *circular dependency* makes it difficult to mitigate our proposed attack easily.

4.2 Technical Challenges

To implement the botnet as designed above, we need to address the following technical challenges. *(1) Bot infection challenge:* A paging request is created only if the recipient UE is registered to the network but idle. Hence, to launch a paging storm attack, the botmaster needs to know if an UE is in such an exploitable state. Such knowledge can be obtained by malware installed on the UE and reported to the botmaster. *(2) State inference challenge:* On a mobile phone, the LTE network protocol stack is typically implemented by its baseband processor. After the bot malware infects the mobile device, it runs only on its application processor. Hence, the LTE network state of the UE can only be inferred through use of telephony APIs (Application Programming Interfaces) provided and permitted by the OS (Operating System) running on the application processor. *(3) State transmission challenge:* If the bot malware detects the LTE network state of the UE to be idle, sending this state to the botmaster through the LTE network requires the UE's reconnection to the network and thus interfere with its idle state in the network, leading to a paradoxical situation! *(4) Attack coordination challenge:* With information collected about idle and connected UEs, the botmaster commands the bots on the connected UEs to

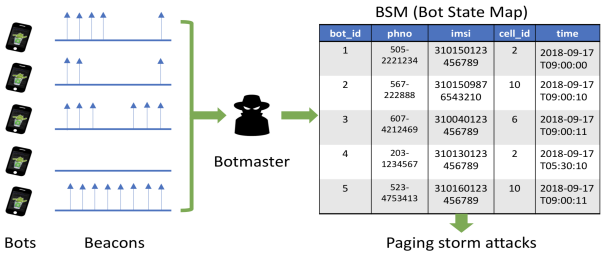


Figure 4: Android botnet for paging storm attacks

send short messages to the idle ones. As in any other pulsating DDoS (Distributed Denial of Service) attacks [30, 38, 39, 44], effective coordination is needed among bot malware to trigger pulsating paging requests in the network.

The *bot infection* challenge can be addressed by traditional malware delivery mechanisms on the Android platform. For example, bot malware can pretend to be a legitimate application in an online marketplace for Android apps, such as Google Play, or a malicious website tricks a visiting mobile device into downloading the bot malware. It is out of the scope of this work to develop a specific bot distribution mechanism for launching paging storm attacks.

4.3 Algorithm description of Android botnet

We now present the algorithms of a proof-of-concept Android botnet which overcomes the aforementioned challenges. Its schematic is illustrated in Figure 4. The practicality of our proposed botnet has been validated on both Android 4.4 (KitKat) and Android Q.

Bot. Each individual bot wakes up periodically to detect if there is any cellular data activity by the UE in the last period. If there are any cellular data received or sent, it means that the UE must not be in an idle state so the bot sends a beacon message to the botmaster, including the current cell where the UE resides. In the first beacon message, the bot also sends the UE's IMSI (International Mobile Subscriber Identity) and phone number to the botmaster. If there is no cellular data activity, the bot does not send a beacon message, which otherwise may interfere with the state of an idle UE. The bot further proceeds to wait for any command message from the botmaster, and if there is any, it sends short messages to a list of phone numbers contained in the message.

The pseudocode of each individual bot is given in Algorithm 1, in which the Android APIs needed by the bot malware are explained at its bottom. The APIs used by the bot to detect if there is any cellular data activity are TrafficStats.getMobileRxBytes() and TrafficStats.getMobileTxBytes(), which return the number of bytes received and sent across the mobile networks since device boot, respectively. We use these two APIs to check whether the total number of bytes received or sent across the mobile networks in the last period is 0 (Line 7 in Algorithm 1), and if it is not 0, a beacon message is sent to the botmaster. It is noted that the transmission of the beacon message itself should be excluded from the measurement; otherwise, there may be always data transmitted across the mobile networks in the last period due to the beacon message. Hence, the size of the beacon message is used to update the counter for transmitted bytes since device boot (Line 18 in Algorithm 1). It is possible that the beacon message is sent *not* across the cellular network (e.g., over the Wi-Fi network) but the counter is still updated by its size. In that case, the last period can be falsely treated as idle if there were no data received over the mobile network but the number of bytes sent by the UE over the cellular network is exactly equal to the size of the beacon message (i.e., the condition for not passing the check on Line 7 in Algorithm 1). Such scenarios are rare and can be ignored for simplicity.

Algorithm 1 Pseudocode for each individual Android bot

1: Call PowerManager.newWakeLock()[a] to keep the bot running in the background
2: $beacon\_id \leftarrow 0$, $old\_rx\_bytes \leftarrow 0$, $old\_tx\_bytes \leftarrow 0$
3: **while** true **do**
4: Call Timer()[b] to schedule a timer that expires after τ time units
5: $rx\_bytes \leftarrow$ TrafficStats.getMobileRxBytes()[c]
6: $tx\_bytes \leftarrow$ TrafficStats.getMobileTxBytes()[d]
7: **if** $rx\_bytes \neq old\_rx\_bytes$ or $tx\_bytes \neq old\_tx\_bytes$ **then**
8: $old\_rx\_bytes \leftarrow rx\_bytes$, $old\_tx\_bytes \leftarrow tx\_bytes$, $msg \leftarrow \emptyset$
9: **if** $beacon\_id == 0$ **then**
10: $imsi \leftarrow$ TelephonyManager.getSubscribeId()[e]
11: $phno \leftarrow$ TelephonyManager.getLine1Number()[f]
12: $msg \leftarrow msg \cup \{imsi, phno\}$
13: **end if**
14: $beacon\_id \leftarrow beacon\_id + 1$, $bot\_id \leftarrow self$
15: $cell\_id \leftarrow$ UE's current cell ID
16: $msg \leftarrow \{bot\_id, cell\_id\} \cup msg$
17: Create a DatagramPacket for msg and send it to the botmaster
18: $old\_tx\_bytes \leftarrow old\_tx\_bytes + sizeof(msg)$
19: **end if**
20: `WAIT:`
21: Wait for any command message from the botmaster **or** for the timer to expire
22: **if** there is any command message from the botmaster **then**
23: $old\_rx\_bytes \leftarrow$ TrafficStats.getMobileRxBytes()
24: $old\_tx\_bytes \leftarrow$ TrafficStats.getMobileTxBytes()
25: **for** each phone number $phno$ in the command message **do**
26: Call SmsManager.getDefault().sendTextMessage()[g] to send a short message to $phno$
27: **end for**
28: **go to** `WAIT`
29: **else if** the timer expires **then**
30: **continue**
31: **end if**
32: **end while**

[a] *PowerManager.newWakeLock()* is called to run malware when the screen of the mobile device is turned off. Permission WAKE_LOCK is needed.
[b] *Timer()* is called for sending periodic beacons. No permission is needed.
[c] *TrafficStats.getMobileRxBytes()* returns the number of received bytes through cellular interface. No permission is needed.
[d] *TrafficStats.getMobileRxBytes()* returns the number of sent bytes through cellular interface. No permission is needed.
[e] *TelephonyManager.getSubscribeId()* gets the IMSI of the device. Permission GET_PHONE_STATE is needed.
[f] *TelephonyManager.getLine1Number()* gets the phone number of the device. Permission GET_PHONE_STATE is needed.
[g] *SmsManager.getDefault().sendTextMessage()* is used to send a short message. Permission SEND_SMS is needed.

Botmaster. The pseudocode of the botmaster is given in Algorithm 2. As shown in Figure 4, it uses a BSM (Bot State Map) to keep the state of each bot-infected UE. The BSM maps from the bot ID to a quadruple that represents the corresponding bot's state, including the phone number, IMSI, the current cell, and the last update time. The botmaster keeps listening to the beacon messages from individual bots. Whenever there is a new beacon message coming, it updates the state of the corresponding bot with the information included in the message and modifies the last update time to be the current time. The botmaster also periodically reads the

Algorithm 2 Pseudocode for the botmaster

```
BSM ← ∅                ▷ BSM (Bot State Map) is a map from a bot ID to a quadruple, indexed from 0 to 3.
Schedule a timer that expires every τ time units
while the timer expires or a new beacon message arrives do
    time ← current time
    if a new beacon message arrives then
        Extract bot_id, cell_id, imsi, and phno from the message
        BSM[bot_id] ← (phno, imsi, cell_id, time)
    else
        idle_set ← ∅, connected_set ← ∅
        for every bot_id in BSM.keys() do
            if t − BSM[bot_id][3] > θ then   ▷ Last beacon seen more than θ time units ago
                idle_set ← {bot_id} ∪ idle_set
            else
                connected_set ← {bot_id} ∪ connected_set
            end if
        end for
        if idle_set and connected_set satisfy paging storm attack condition then
            Send a command message to each bot in connected_set, which includes a partial list of phone numbers associated with the bots in idle_set
        end if
    end if
end while
```

BSM and updates two sets, one including those likely to be in an idle state (*idle_set*) and the other those likely to be in a connected state (*connected_set*). Using these two sets, the botmaster decides if it needs to command the bots on connected devices to send short messages to the idle ones.

The botmaster can use the cell information reported by each bot to decide which cell to be attacked. If a target cell is chosen for the attack, short messages will be sent to only those devices with matched *cell_id*'s in the *idle_set*. We assume that the botmaster attacks a target cell only if the number of idle UEs inside the cell exceeds a certain threshold (e.g., 100 idle UEs). As we do not assume that the botmaster's machine can send out short messages, it commands the bots in the *connected_set* to send short messages to the UEs in the *idle_set*. The botmaster can repeat this attack whenever the attack condition holds.

It is possible that *idle_set* may include those UEs that have been powered off or whose cellular services have been tured off because the bot malware on these devices cannot send beacons to the botmaster either. A short message delivered to such a UE does not trigger an immediate paging request because it has been detached from the network. As the size of *idle_set* may overestimate the number of UEs in idle states, the botmaster can adjust threshold θ upward accordingly to ensure that there are sufficient idle UEs to cause pulsating paging requests within a short period of time.

5 EMULATION TESTBED

Due to ethical issues we do not evaluate the attack effects of the Android botnet on an operational LTE network. Instead, we develop a high-fidelity emulation testbed for this purpose.

5.1 Testbed Components

LTE network emulation. We use `OpenAirInterface` [8] for LTE network emulation. OpenAirInterface includes `openairinterface5g` to emulate eNodes and UEs, and `openair-cn` to emulate the EPC. The master branch of OpenAirInterface lacks three key features needed to study the effects of paging storm attacks: (1) *Handling of paging requests:* a paging request initiated by the MME inside the EPC is broadcast to all the eNodeBs in the tracking area of the recipient UE, which further broadcast the paging request at the appropriate paging occasions. (2) *RRC inactivity timer:* An eNodeB has an inactivity timer scheduled for each UE that it is currently serving at the RRC layer. If there is no data traffic between the UE and the eNodeB before the timer expires, the eNodeB initiates an RRC release procedure to release the RRC connection between the UE and the eNodeB. (3) *SMS support:* There are two types of SMS support SMS in LTE: SMS over SGs and SMS over IMS (IP Multimedia Subsystem).

In our implementation, we ported the first two features from the `develop` branches of `OpenAirInterface`. To support short message delivery, we implement the SMS-over-SGs mechanism due to its simplicity. We created additional entities in `openair-cn` according to the 3GPP specification for SMS over SGs [11].

Android device emulation. The bot malware in the proof-of-concept Android botnet is implemented as an Android APK, executed by the official Android Emulator [1]. The Android Emulator uses a `qemud` daemon [9] to manage multiple AVDs (Android Virtual Devices), each allowed to connect to the different services (e.g., GPS, sensors, and GSM) through universal interfaces. To support phone calls and SMS, the GSM service implements a communication channel through which GSM AT commands are exchanged between the Android OS and an emulated modem in the AVD. For example, an AVD uses the `AT+CMGS` and `AT+CMGR` command to send and receive short messages, respectively. To exchange AT commands between the GSM service and the emulated modem, the Android emulator uses `CharPipe`, which works like a normal Unix pipe. We intercept the AT commands inside the Android Emulator and forward them to the AT command interface of the UE Emulator in `openairinterface5g`. We hook the CharPipe data structure from the modem side due to the complexity of functionalities on the GSM service side. As both the Android Emulator and the UE Emulator run on the same physical machine in our testbed, we use a simple UDP socket to communicate the AT commands between them. Inside the UE Emulator, AT commands received from the Android Emulator are forwarded to the NAS layer of the UE.

User activity simulator. The idleness of a device hinges upon its user activity model. As the attack effects are only affected by infected UEs' idleness, we use a simple ON/OFF model to simulate user activities on each emulated Android device. In an OFF state, the user is assumed to be idle, and in an ON state, it is assumed that the user sends periodic `ping` traffic to an external server. The ON/OFF traffic model is implemented as an Android APK running along with the bot malware in our testbed. Human dynamics exhibit heavy tails in various situations [13]. As it has been shown that the heavy-tailed Pareto distribution can be used to model user data collected from real-world cellular networks [32], we use it to model the lengths of both the ON and OFF states in the traffic model. The heavy-tailed Pareto distribution, $F(x) = 1 - (m/x)^{\alpha}$, has two parameters, scale parameter m and shape parameter α.

5.2 Testbed Realism

To ensure that the evaluation results from the testbed should be realistic, we have considered the following issues.

First, as the mobile bot is executed within the official Android emulator, the software execution overhead experienced by its network traffic is realistic. In the emulation testbed, the LTE modem is emulated by the network protocol stack of the UE emulator. Although the software emulation delay may *overshoot* the processing delay of the LTE modem in a real mobile phone, it should be much smaller than the duration of a DRX cycle (0.32 second in our experiments). Moreover, the shorter processing delays of the LTE modems in real mobile phones actually help pulsating paging storm attacks. Hence, using emulated LTE modems may underestimate the attack effects.

Second, in OpenAirInterface the base station and the core network elements are each emulated by separate threads. Across different protocol layers the messages are sent and received using the intertask interfaces with messages stored in queues. These queues are implemented based on the liblfds library, whose users include AT&T, Red Hat and Xen [2]. Hence, the OpenAirInterface LTE network emulator has used production-ready software libraries in its implementation. Moreover, as all the Android emulators and the OpenAirInterface LTE network emulator run on a cluster machine in our emulation testbed (see Section 6), we also use the netem utility [3] to emulate transmission delays and jitters among these different components. More specifically, we add 50ms delay and 5ms jitter to packet transmissions by the botmaster machine, 10ms delay and 1ms jitter to the link between the eNodeB and MME as well as S/PGW, and 20ms delay and 2ms jitter to the link between the MME and the SMS center. We choose the transmission delays at the order of tens of milliseconds based on measurement results from the real-world AT&T network [4].

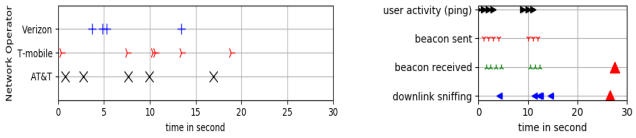
Last, we minimize the artifacts in delay measurements in our testbed due to contention of computational resources. The memory requirement of each AVD client is significant. We configure each AVD client to run Android 4.4 (KitKat) with 512 MBytes of RAM. To reduce CPU and memory usage, we remove the Android emulator's Google Apps and SMS notification. It is noted that the maximum number of concurrent threads that can be executed on our cluster machine is only 72, which is smaller than the number of bot-infected devices in our experiments in Section 6. However, both the Android emulator and the LTE network emulator are *event-driven*: if no activity occurs on an emulated Android device or an emulated LTE network entity, it incurs little computational overhead.

6 EVALUATION

In this section, we first obtain measurements from real LTE networks to validate the botnet design and configure the emulation parameters. We next use the emulation testbed to study the attack effects. For emulation experiments, we use a cluster machine that has two Intel Xeon 6140 processors – thus 36 cores with hyper-threading – and 192GB memory in total with six 32GB DDR4 RAMs.

6.1 Measurements

RRC inactivity timeout. The RRC inactivity timeout value has prominent impact on the signaling overhead and its setting varies with each LTE network operator [12]. To infer practical RRC inactivity timeout values implemented by LTE network operators, we set up an SDR (Software Defined Radio)-based sniffer modified from srsLTE [20] to capture the messages destined to a target UE. On detection of an idle UE, the eNodeB sends an RRC connection release message to the UE. As non-broadcast messages transmitted at the RRC layer are encrypted, we cannot explicitly filter out the RRC connection release messages sent to the target UE without knowing the encryption key. We thus infer its existence by monitoring the last message destined to the TMSI (Temporary Mobile Subscriber Identity) of the target UE because, without any new data request from or to the target UE, no message interaction should occur at the RRC layer after the eNodeB sends the RRC connection release message to the UE. To force the RRC release procedure, we pull out the SIM card of a smart phone and immediately start a timer that expires when the SDR-based sniffer captures the last message destined to its TMSI. When the timer expires, we measure its duration as an estimate of the RRC inactivity timeout value.



(1) RRC timeout estimation (2) Idleness inference

Figure 5: Measurements from real 4G/LTE networks

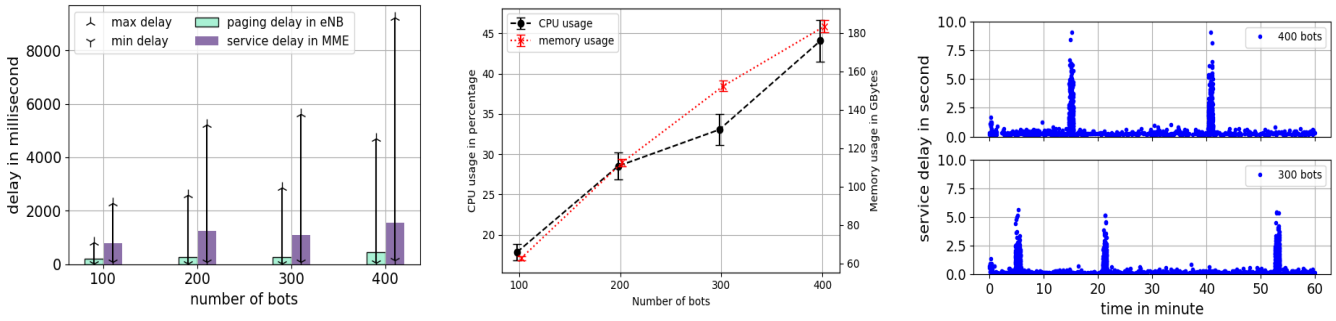
We did experiments for three cellular network operators in our local area: AT&T, T-mobile and Verizon. As shown in Figure 5(1), the last control plane message is received after about 14 to 18 seconds for each of these network operators. Based on this observation, we configure the RRC timeout value in the emulation experiments to be 15 seconds; we also assume that the botmaster infers the idleness of a UE from which it has not received a beacon message for more than 15 seconds (parameter θ in Algorithm 2).

Idleness inference algorithm. We perform measurements to evaluate the effectiveness of inferring a UE's idleness by the botmaster. The botmaster scans the BSM every 15 seconds. On a real Android phone connected to the Verizon network, we execute the bot malware along with the user activity simulator.

Figure 5(2) shows the time at which the user activity simulator sends a ping packet (1^{st} line), the bot malware sends a beacon message (2^{nd} line), the botmaster receives the beacon message (3^{rd} line), and the SDR-based sniffer detects a message destined to the UE (4^{th} line). In this example, the botmaster detects the UE to be idle at the time shown as a upward triangle on the 3^{rd} line, and the sniffer detects the last downlink packet delivered to the UE at the time shown as a upward triangle on the 4^{th} line. Because the upward triangle on the 3^{rd} line occurs *after* the one on the 4^{th} line, the network should have already released its resources for the UE when the botmaster infers that the UE is in an idle state.

Size estimation of regional botnets. We use the Town of Los Alamos in the New Mexico State of US as an example to estimate the possible size of a regional botnet[1]. Los Alamos has a population of 12,019, according to its wikipedia page [52]. It has four cellular network operators, Sprint, T-Mobile, Verizon, and AT&T; as it has four cell towers, we can assume that each cellular network operator has a single tower in Los Alamos [16]. As reported in the GSMA

[1]Los Alamos is a small isolated town, which makes it easy for botnet size estimation. We believe that the denser population and thus denser mobile devices in metropolitan areas should make paging storm attacks even more effective.



(1) Attack effects of botnet size (2) CPU usage and memory footprint (3) Service delay in MME over time

Figure 6: Attack effects of botnet size, CPU and memory overhead, and pulsation of paging requests

Mobile Economy Report, the subscriber penetration rate in North America is 83%, among which the use rate of 4G technology is 69% in 2018 [22]. Moreover, the market share of Android in US is 42.75% in December 2018 according to [27]. Using the average market share of each cellular network operator in US (T-mobile 14%, Sprint 17%, Verizon 31%, and AT&T 34%) [46] and a normalization factor of 1.042 (i.e., 1.0 / (0.14 + 0.17 + 0.31 + 0.34)), we can extrapolate the number of Android users in the Los Alamos area per cellular network operator: T-mobile 429, Sprint 521, Verizon 951, and AT&T 1042. If we assume that a malware program pretending to be a localized mobile application has a penetration rate of 40%, a regional botnet can have as many as 400 bots. Based on our estimation, we vary the size of a regional botnet between 100 and 400. To validate the existence of popular regional mobile apps in the Los Alamos area, we search keyword "Los Alamos" in Google Play and find more than 50 Android apps targeting the town of Los Alamos. The "Los Alamos Trails" app has had more than a thousand installations [6].

6.2 Emulation experiments

Table 2: Default parameter settings in our experiments

Parameter	Meaning	Default
T	Length of a DRX cycle	32 frames
B	Number of paging occasions per frame	$1T$
R	Length of paging record list	7
U_b	Number of bot-infected UEs	300
U_l	Number of clean UEs	0
(m, α)	Pareto distribution parameters	(15, 3)

We use the default values in Table 2 to set the parameters in the experiments. We evaluate the attack effects from a regional botnet with a few hundred bots sharing the same eNodeB but do not emulate clean UEs to minimize the computational overhead while still gaining insights into the attack effects. For simplicity, we use the same Pareto distribution parameters for both ON and OFF states in the user activity simulation model, and their mean and standard deviation are 22.5 and 12 seconds, respectively. The duration of each emulation experiment is one hour, and the condition to trigger a new paging storm attack by the botmaster is that the number of idle bot-infected devices exceeds half of the botnet size.

Baseline case without attack. We measure the attack effects as both the *paging delay in eNB*, defined to be the difference between the receiving time and the serving time of a paging message handled by the eNodeB, and the *service delay in MME*, which is the difference between the time when a paging request is sent by the MME to the eNodeB and the time when a service request message is received from the eNodeB by the MME. When there is *no botnet attack*, the delays experienced by a normal paging request are measured as follows: the mean and maximum paging delay in eNB is 172.8 and 412.3 milliseconds, respectively, while the mean and maximum service delay in MME is 452.3 and 849.6 milliseconds, respectively.

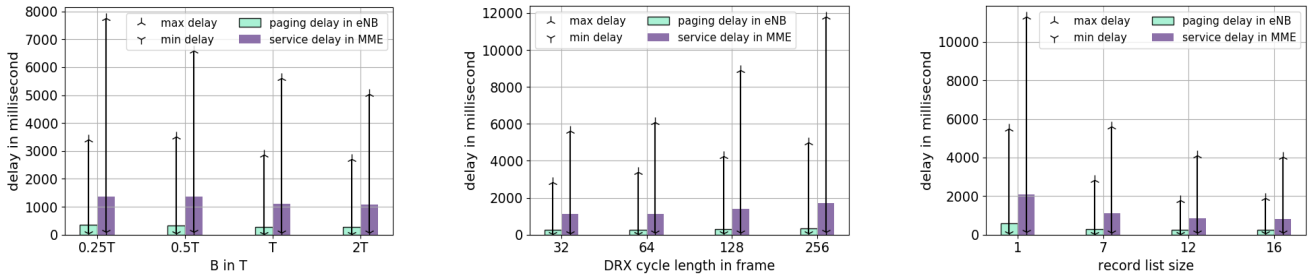
Next we perform a set of experiments to study the attack effects in different scenarios. In each experiment, the parameters are set as in Table 2 except the one being varied. When measuring the attack effects, we only consider those paging or service requests that are started within 30 seconds after a paging storm attack is launched.

B.1: Effects of botnet size and pulsation of paging requests

Effects of botnet size. Varying the botnet size from 100 to 400, the attack effects are shown in Figure 6(1). We see that both the paging delay in eNB and the service delay in MME increase with the botnet size. With more bots, there is a higher probability of collision for paging messages at each paging occasion, which postpones the serving time of the collided paging message. Hence, the emulation results agree with our mathematical analysis in Section 3.

From a mobile user's perspective, the attack effect is manifested by the service delay in MME, because it affects how much time a legitimate phone or video call can be postponed by the paging storm attack, let alone the delays that the botnet attack creates at the other places in the network. From Figure 6(1), we observe that when the botnet infects 200 UEs, the average service delay in MME is more than one second while in the worst case the service delay in MME can be more than five seconds; if the botnet size increases to 400, the service delay in MME can be more than nine seconds.

CPU and memory usage. To understand how the computational resource is used, we depict in Figure 6(2) the CPU usage of each core and the total memory footprint, measured by their means and standard deviations in the emulation experiments. We observe that the CPU and memory utilization levels both grow almost linearly with the number of bots in the experiments. Even when the botnet has 400 bots, the mean CPU utilization level is only around 45%, suggesting that resource contention should have little effect on the delay measurements seen in Figure 6(1).



(1) Parameter B (2) DRX cycle length (3) Record list size

Figure 7: Attack effects of LTE network parameters

Pulsation of paging requests. Figure 6(3) presents the service delay in MME of each individual paging request in a sample run lasting one hour. When the botnet size is 400 (300), there are two (three) paging storm attacks launched by the botmaster, each triggered because there are at least 200 (150) idle bots inferred by the botmaster algorithm (see Algorithm 2). When the botnet size is 400 (300), among all the service requests started within 30 seconds after a paging storm attack is launched, 1.63% (0.79%) of them are delayed by more than 4 seconds, 9.69% (8.91%) by 2 to 4 seconds, and 41.37% (30.49%) by 1 to 2 seconds. If such attacks are repeatedly performed, a large number of legitimate calls can be affected.

In our experiments, we measure the attack effects by considering those service requests started within 30 seconds after a page storm attack is launched. If we vary the time cutoff among 5, 15, 30, 45, and 60 seconds for the scenario with 400 bots in Figure 6(3), we get the average service delay in MME to be 2.41, 2.16, 1.91, 1.85, and 1.80 seconds, respectively. The decreasing service delay suggests that the impact of a paging storm attack on a later service request decreases with its starting time, which agrees well with our intuition.

B.2: Effects of LTE network parameters

Effects of parameter B. Parameter B gives the number of paging occasions in each frame. For example, when B equals $T/4$, one paging occasion occurs every four frames; similarly, having B equal to $2T$ means each frame has two paging occasions to serve paging messages. Obviously the larger B is, the larger capacity the eNodeB has to process paging requests, leading to smaller paging and service delays due to fewer collisions among the paging messages. In our experiments we evaluate four different settings for parameter B: $T/4$, $T/2$, T, and $2T$. Figure 7(1) shows how the paging delay in eNB and the service delay in MME vary with parameter B. We observe that when B is $T/4$, the service delay in MME can be as high as almost eight seconds. A higher value of parameter B increases the eNodeB's capacity for delivering the paging requests to UEs per DRX cycle, thus reducing both the average paging delay in eNB and the average service delay in MME. Still, we see that the service delay can be close to five seconds even if $B = 2T$.

Effects of parameter T. Parameter T decides the duration of each DRX cycle, which affects the frequency at which a UE wakes up periodically to receive paging messages in an `RRC-IDLE` state. Intuitively, a shorter DRX cycle means that a UE wakes up more frequently, thus shortening the service time of a paging request at the cost of higher power consumption. On the other hand, a larger DRX cycle increases the service time of each paging request while reducing the UE's power consumption. In our experiments, we consider the four typical lengths of DRX cycles according to the 3GPP specifications: 32, 64, 128, and 256 frames.

The experimental results shown in Figure 7(2) confirm our intuition above. Having a longer DRX cycle slows down the delivery of paging requests to the recipient UEs, thus increasing both the paging delay in eNB and the service delay in MME. It is noted that when the length of a DRX cycle is 32 or 64 frames, the maximum service delay in MME can be close to six seconds, and when each DRX cycle has 256 frames, the maximum service delay in MME gets close to twelve seconds. Hence, in LTE networks with long DRX cycles, a paging storm attack can significantly increase the call connection setup time.

Effects of parameter R. When multiple paging messages are hashed to the same paging occasion, they can be delivered in the same paging record list. Parameter R decides the size of each paging record list with a maximum value of 16. When the paging record list is full, paging messages hashed to the same paging occasion have to be postponed to the next DRX cycle. We consider four different settings for parameter R: 1, 7, 12, and 16.

Figure 7(3) illustrates the paging delay in eNB and the service delay in MME under different sizes of paging record lists. When the paging record list is as short as containing at most one paging message, the average service delay in MME exceeds two seconds and its maximum delay exceeds eleven seconds. When we increase the size of the paging record list, both the average paging delay in eNB and the average service delay in MME tend to decrease. These observations agree well with our intuition that a longer paging record list should allow more paging messages to be delivered at the same paging occasion, thus reducing the average service time of each paging request by the network. Still even with $R = 16$, the average service delay in MME is close to one second and its maximum delay is as long as four seconds.

7 DISCUSSIONS

In this section we discuss the practicality of paging storm attacks, limitations of this work, and potential mitigation schemes.

Attack practicality: In our experiments, a paging storm attack is launched when at least half of the bots in the botnet are inferred as idle. When the penetration rate of the bot malware is lower, the botmaster can use a higher threshold to trigger a new attack. For example, with a botnet of 150 bots, the botmater can launch a new

paging storm attack when 130 of the bots are found idle. Moreover, it is also possible to use other types of LTE-capable devices in a paging storm attack (e.g., iOS devices and IoT devices).

Using a regional botnet, it is possible to perform other types of attacks against 4G/LTE networks. For example, bot malware can make simultaneous service requests to cause pulsating signalling messages in the network. An idle UE initiating a service request first establishes an RRC connection with an eNodeB and then makes a service request to the MME [37]. If the majority of the infected devices are in an idle state, the attack can congest the shared common control channels used for RRC connection establishment. Similar to a paging storm attack, such an attack also needs inference about idleness of UEs and coordination among bot malware to ensure that bot-generated service requests occur within a short period of time. Extending our testbed to study the effects of different types of attacks against cellular networks remains as our future work.

Limitations. The consequences of paging storm attacks are limited to delaying incoming calls by a few seconds; they are not likely to cause service disruption of cellular networks nor have significant impact on many people's lives which can tolerate delayed calls. To further maximize the attack effects, the botmaster can adopt a salami attack tactic [23, 29] by performing repetitive paging storm attacks under the radar over a long period of time.

Despite our great efforts on developing the high-fidelity emulation testbed, some factors were ignored. For example, only a single eNodeB was considered but there may be multiple eNodeBs in the same area. However, paging storm attacks can still be effective because paging is performed at the track area level: a paging request from an MME is sent to all the eNodeBs in the same track area. Due to limited computational resources, we did not consider any legitimate paging requests to uninfected mobile phones, but their existences should magnify the attack effects. A user may power off her mobile phone, change her phone to an airplane mode, or use WiFi instead of LTE. We plan to enrich our emulation testbed by taking these factors into account in our future work.

Mitigation. Potential mitigation schemes against paging storm attacks include the following. First, the periodic beacon messages sent by each bot to the botmaster's machine can be leveraged for botnet detection. This scheme may however generate some false alarms. Second, as each bot infers the UE's idleness by measuring cellular data activities, we can use a proxy thread on the device to break its idleness by sending some data over the cellular network. This countermeasure, however, not only wastes power and increases the cellular data use of the mobile phone but also forces the cellular network to keep the resources for an actually idle mobile phone. Third, as seen in Figure 7, the cellular network operator can increase parameter B, reduce the DRX cycle length, or increase the record list size to mitigate the effects of paging storm attacks. But there is no free lunch here: increasing B means fewer slots for other channels not used for paging in a frame, reducing the DRX cycle length means that each mobile phone has to wake up more frequently to check paging requests and its battery thus drains more rapidly, and the record list size can be at most 16 in LTE.

8 RELATED WORK

This section presents previous works related to vulnerabilities of 4G/LTE networks and attacks on cellular networks.

Vulnerability assessment of 4G/LTE networks. Jover surveyed various attacks against the availability of LTE networks [28]. In another survey Rupprechet *et al.* discussed the security challenges of different generations of mobile communication networks based on their root causes [40]. Shaik *et al.* demonstrated practical attacks against several vulnerabilities in the LTE access network protocol specifications [43]. Tu *et al.* investigated the vulnerabilities resulting from problematic protocol interactions [51]. Hong *et al.* analyzed GUTI (Globally Unique Temporary Indentifier) reallocation data from tens of carriers and found patterns that could compromise subscribers' privacy [24]. Rupprecht *et al.* identified three attack vectors at the layer two in the LTE network protocol stack that may compromise communication confidentiality or privacy [41]. Hussain *et al.* proposed to use a combination of symbolic model checking and cryptographic protocol verifier to identify vulnerabilities in LTE network protocols [25]. Fang and Yan used reinforcement learning to assist with discovery of vulnerabilities in LTE networks [19]. Kim *et al.* developed methods to generate test cases for assessing the vulnerabilities of the LTE control plane [31]. Hussain *et al.* found that mobile users' private information can be revealed from paging messages via side channel attacks [26].

Large-scale attacks on cellular networks. Enck *et al.* investigated the security risk of SMS interface to cellular networks and the feasibility of attacking a cellular network of a national scale with a medium-sized botnet [18]. Traynor *et al.* studied the possibility of using a cellular botnet to overload the HLR (Home Location Register) in 2G or 3G networks [49]. Lee *et al.* proposed to use CUSUM (cumulative sum) test to detect signaling DoS (Denial of Service) attacks against 3G networks [34]. Tu *et al.* analyzed the security threats caused by IMS-based SMS service in 4G/LTE networks and performed feasibility study of large-scale attacks [50]. Bassil *et al.* proposed a signaling-oriented DoS attacks against LTE networks by exploiting the numerous signaling exchanges when setting up a dedicated bearer in an LTE network [14]. Serror *et al.* has suggested that attacks from the Internet can overload a CDMA2000 cellular network with large amounts of paging messages, leading to increased delay of cellular call setup requests [42]. Traynor *et al.* considered jamming attacks against the Random Access channels in GSM networks and studied their impact on normal traffic [48].

9 CONCLUSIONS

In this work we study the attack effects of paging storm attacks from regional botnets. We implement a proof-of-concept Android botnet for such attacks. To understand its attack effects, we mathematically analyze the probability of delaying paging requests and built a high-fidelity emulation testbed connecting emulated Android devices and an LTE network emulator. We use measurement results from real-world LTE networks to validate our botnet design and perform intensive emulation experiments to shed light on the attack effects in a practical environment. Our experiments show that paging storm attacks launched from a regional botnet can delay time-critical voice/video calls by several seconds.

ACKNOWLEDGMENTS

We acknowledge the Critical Infrastructure Resilience Institute, a US DHS Center of Excellence, for supporting this work. We also thank anonymous reviewers for their valuable comments.

REFERENCES

[1] *https://developer.android.com/studio/run/emulator*.
[2] *https://liblfds.org*.
[3] *https://wiki.linuxfoundation.org/networking/netem*.
[4] *http://ipnetwork.bgtmo.ip.att.net/pws/network_delay.html*.
[5] https://www.callcentrehelper.com/industry-standards-metrics-125584.htm. *https://www.callcentrehelper.com/industry-standards-metrics-125584.htm*.
[6] Los alamos trails. *https://play.google.com/store/apps/details?id=org.pajaritoeec.losalamostrailsapp*.
[7] LTE Quick Reference: SIB(System Information Block) Modification/Notification. *https://www.sharetechnote.com/html/Handbook_LTE_SIB_Modification.html*.
[8] OpenAirInterface. *http://www.openairinterface.org/*.
[9] Qemu. *https://www.qemu.org/*.
[10] What causes a delay in delivering sms messages? *https://help.nexmo.com/hc/en-us/articles/204014893-What-Causes-a-Delay-in-Delivering-SMS-Messages-*.
[11] 3GPP TS 23.272 V11.9.0. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Circuit Switched (CS) fallback in Evolved Packet System (EPS); Stage 2 (Release 11). 2014.
[12] 3GPP TS 36.822 V11.0.0. 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; LTE Radio Access Network (RAN) enhancements for diverse data applications (Release 11). 2012.
[13] A.-L. Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435(7039):207, 2005.
[14] R. Bassil, A. Chehab, I. Elhajj, and A. Kayssi. Signaling oriented denial of service on LTE networks. In *Proceedings of the 10th ACM international symposium on Mobility management and wireless access*, pages 153–158. ACM, 2012.
[15] Business Insider Intelligence. App localization increases app ROI. *https://www.businessinsider.com/app-localization-increases-app-roi-2016-5*, 2016.
[16] Cellreception. Los alamos, nm cell towers & signal map. *http://www.cellreception.com/towers/towers.php?city=los+alamos&state_abr=nm*.
[17] W. Chen, X. Luo, C. Yin, B. Xiao, M. H. Au, and Y. Tang. MUSE: towards robust and stealthy mobile botnets via multiple message push services. In *Australasian Conference on Information Security and Privacy*, pages 20–39. Springer, 2016.
[18] W. Enck, P. Traynor, P. McDaniel, and T. La Porta. Exploiting open functionality in sms-capable cellular networks. In *Proceedings of the 12th ACM conference on Computer and communications security*, pages 393–404. ACM, 2005.
[19] K. Fang and G. Yan. Emulation-instrumented fuzz testing of 4G/LTE android mobile devices guided by reinforcement learning. In *European Symposium on Research in Computer Security*, pages 20–40. Springer, 2018.
[20] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith. srsLTE: An open-source platform for LTE evolution and experimentation. *arXiv preprint arXiv:1602.04629*, 2016.
[21] GSMA. Network 2020: Mission critical communications. *https://www.gsma.com/futurenetworks/wp-content/uploads/2017/03/Network_2020_Mission_critical_communications.pdf*.
[22] GSMA. The mobile economy. *https://www.gsma.com/r/mobileeconomy/*, 2019.
[23] B. M. Hale. Salami attacks. *http://all.net/CID/Attack/papers/Salami2.html. Accessed in December 2019*.
[24] B. Hong, S. Bae, and Y. Kim. GUTI reallocation demystified: Cellular location tracking with changing temporary identifier. In *Proceedings of Symposium on Network and Distributed System Security*. Internet Society, 2018.
[25] S. R. Hussain, O. Chowdhury, S. Mehnaz, and E. Bertino. LTEInspector: A systematic approach for adversarial testing of 4G LTE. *Proceedings of the Network and Distributed System Security Symposium*, 2018.
[26] S. R. Hussain, M. Echeverria, O. Chowdhury, N. Li, and E. Bertino. Privacy attacks to the 4G and 5G cellular paging protocols using side channel information. In *Proceedings of The 26th Network and Distributed System Security Symposium (NDSS'19)*, 2019.
[27] S. Ireland. *https://ceoworld.biz/2019/01/17/most-popular-mobile-operating-systems-in-the-united-states-android-vs-ios-market-share-2012-2018/*, 2019.
[28] R. P. Jover. Security attacks against the availability of LTE mobility networks: Overview and research directions. In *Wireless Personal Multimedia Communications (WPMC)*. IEEE, 2013.
[29] M. Kabay. Salami fraud. *Network World Security Newsletter*, 24, 2002.
[30] Y.-M. Ke, C.-W. Chen, H.-C. Hsiao, A. Perrig, and V. Sekar. Cicadas: congesting the internet with coordinated and decentralized pulsating attacks. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 699–710. ACM, 2016.
[31] H. Kim, J. Lee, E. Lee, and Y. Kim. Touching the untouchables: Dynamic security analysis of the LTE control plane. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 2019.
[32] M. Laner, P. Svoboda, S. Schwarz, and M. Rupp. Users in cells: a data traffic analysis. In *Wireless Communications and Networking Conference*. IEEE, 2012.
[33] H. Lee, T. Kang, S. Lee, J. Kim, and Y. Kim. Punobot: Mobile botnet using push notification service in Android. In *International workshop on information security applications*, pages 124–137. Springer, 2013.
[34] P. P. Lee, T. Bu, and T. Woo. On the detection of signaling DoS attacks on 3G wireless networks. In *Proceedings of the 26th IEEE International Conference on Computer Communications*. IEEE, 2007.
[35] C. Lever, M. Antonakakis, B. Reaves, P. Traynor, and W. Lee. The core of the matter: Analyzing malicious traffic in cellular carriers. In *Proceedings of Symposium on Network and Distributed System Security*, 2013.
[36] C. Mulliner and J.-P. Seifert. Rise of the ibots: Owning a telco network. In *Proceedings of the International Conference on Malicious and Unwanted Software*, pages 71–80. IEEE, 2010.
[37] M. Olsson, C. Mulligan, S. Sultana, S. Rommer, and L. Frid. *EPC and 4G packet networks: driving the mobile broadband revolution*. Academic Press, 2013.
[38] J. Park, D. Nyang, and A. Mohaisen. Timing is almost everything: Realistic evaluation of the very short intermittent ddos attacks. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–10. IEEE, 2018.
[39] R. Rasti, M. Murthy, N. Weaver, and V. Paxson. Temporal lensing and its application in pulsing denial-of-service attacks. In *2015 IEEE Symposium on Security and Privacy*, pages 187–198. IEEE, 2015.
[40] D. Rupprecht, A. Dabrowski, T. Holz, E. Weippl, and C. Pöpper. On security research towards future mobile network generations. *IEEE Communications Surveys & Tutorials*, 20(3):2518–2542.
[41] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper. Breaking LTE on layer two. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 2019.
[42] J. Serror, H. Zang, and J. C. Bolot. Impact of paging channel overloads or attacks on a cellular network. *Proceedings of the 5th ACM workshop on Wireless security*, pages 75–84, 2006.
[43] A. Shaik, R. Borgaonkar, N. Asokan, V. Niemi, and J.-P. Seifert. Practical attacks against privacy and availability in 4G/LTE mobile communication systems. *Proceedings of the Network and Distributed System Security Symposium*, 2015.
[44] H. Shan, Q. Wang, and Q. Yan. Very short intermittent DDoS attacks in an unsaturated system. In *International Conference on Security and Privacy in Communication Systems*, pages 45–66. Springer, 2017.
[45] K. Singh, S. Sangal, N. Jain, P. Traynor, and W. Lee. Evaluating Bluetooth as a medium for botnet command and control. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2010.
[46] Statista. Wireless subscriptions market share by carrier in the u.s. from 1st quarter 2011 to 3rd quarter 2019. *https://www.statista.com/statistics/199359/market-share-of-wireless-carriers-in-the-us-by-subscriptions/*.
[47] K. Suzuki, D. Tonien, K. Kurosawa, and K. Toyota. Birthday paradox for multi-collisions. In *Information security and cryptology–icisc 2006*. Springer, 2006.
[48] P. Traynor, C. Amrutkar, V. Rao, T. Jaeger, P. McDaniel, and T. La Porta. From mobile phones to responsible devices. *Security and Communication Networks*, 4(6):719–726, 2011.
[49] P. Traynor, M. Lin, M. Ongtang, V. Rao, T. Jaeger, P. McDaniel, and T. La Porta. On cellular botnets: measuring the impact of malicious devices on a cellular network core. In *ACM conference on Computer and communications security*, 2009.
[50] G.-H. Tu, C.-Y. Li, C. Peng, Y. Li, and S. Lu. New security threats caused by IMS-based SMS service in 4G LTE networks. In *Proceedings of the 2016 ACM Conference on Computer and Communications Security*. ACM, 2016.
[51] G.-H. Tu, Y. Li, C. Peng, C.-Y. Li, H. Wang, and S. Lu. Control-plane protocol interactions in cellular networks. *ACM SIGCOMM Computer Communication Review*, 44(4):223–234, 2015.
[52] Wikipedia. Los Alamos, New Mexico. *https://en.wikipedia.org/wiki/Los_Alamos,_New_Mexico. Accessed in December 2019*.
[53] Y. Zeng, K. G. Shin, and X. Hu. Design of SMS commanded-and-controlled and P2P-structured mobile botnets. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, pages 137–148. ACM, 2012.