Exploiting Delay Patterns for User IPs Identification in Cellular Networks

Vasile C. Perta, Marco V. Barbera, and Alessandro Mei

Sapienza University, Rome, Italy
{perta,barbera,mei}@di.uniroma1.it

Abstract. A surprisingly high number of mobile carriers worldwide do not block unsolicited traffic from reaching their mobile devices from the open Internet or from within the cellular network. This exposes mobile users to a class of low-resource attacks that could compromise their privacy and security. In this work we describe a methodology that allows an adversary to identify a victim device in the cellular network by just sending messages to its user through one or more messaging apps available today on the mobile market. By leveraging network delays produced by mobile devices in different radio states and the timeliness of push notifications, we experimentally show how our methodology is able to quickly identify the target device within 20 messages in the worst case through measurements on a large mobile network.

Keywords: Cellular Networks, Security, Privacy

1 Introduction

The shift from a peer-to-peer to a cloud-based, centralized communication model has made todays mobile devices less exposed to many of the network security threats that characterise desktop computers, such as intrusions through vulnerable listening services. This might be one of the reasons why an unexpectedly high number of cellular network carriers do not block unsolicited traffic from reaching their devices either from the open Internet, or from within the cellular network itself [18]. Unsurprisingly, this configuration is far from secure, and could be exploited to provide harm to the network [18], or, most importantly, to compromise the privacy and security of the end users. For instance, by monitoring the characteristics of the Internet path towards a mobile device, the user location may be tracked in a fine-grained way [21]. Using a "stealth-spam" attack like that described by Peng *et al.* [14], instead, an adversary may quickly drain the user device battery or data plan with a simple stream of UDP packets. Finally, an accurate characterization of RRC radio states of the device [17] could be leveraged to monitor the Internet traffic patterns of the user. Interestingly, these attacks do not necessarily require a powerful adversary, but can be launched by virtually anyone who either knows the IP of the victim user device in the cellular network, or, at least, can individuate a small set of candidate IPs. Given the wide range of IPs that may be assigned to a device, this may sound like a very strong requirement at first. Actually, in many cases it is not. In this paper, we describe a methodology that allows to leverage push-notification services to detect the user IP address in a cellular network.

Our main contributions are the following.

- We define a lightweight methodology for matching users with their IPs in cellular networks. The methodology leverages the delay patterns produced at different radio states together with the near-real-time characteristics of push-based services. To the best of our knowledge, we are the first to show how network delays on cellular networks constitute an effective side-channel that can be used to undermine user privacy.
- We show how our methodology works with the most popular instant messaging apps and is robust with respect to various network and signal-strength conditions.
- We give a precise evaluation of the amount of resources the methodology requires, both in terms of bandwidth and number of instant messages.
- We experimentally validate our methodology through measurements on over 260K IPs of a large cellular network and show that it is able to correctly individuate the user device IP with less than 20 messages.

The rest of this paper is organised as follows. Section 2 introduces the attacker model. In Section 3 we show how network delays can be used as a side channel to infer the recent network activity on a remote mobile device. In Section 4, we describe our IP detection methodology, and present the experimental results in Section 5. Section 6 discusses the feasibility of our methodology. Related work and future research directions are presented in Section 7 and Section 8.

2 Attacker Model

We are interested in knowing whether an adversary can detect, in a cellular network, the IP address of the mobile device of a given user. Note that the IP could be either private or public, depending on whether the operator deployed NAT or not. In the case NAT is used, the attack has to be carried out from within the cellular network. If the adversary owns a popular website, mobile app, or cloud-based service, obtaining the user IP may be trivial. In fact, such an adversary has a larger number of options to violate end-users privacy and security, and falls outside the scope of this paper. In our scenario, instead, the adversary is a malicious small entity, or even a single person that is not necessarily trusted by the user, but that, at the same time, the user does not perceive as a particular privacy or security threat because of its apparently limited power. This model, which is similar to that assumed by Le Blond *et al.* [11], includes people in the user social circle (*e.g.,* friends, coworkers) or entities such as the user employer. These are weak adversaries potentially interested in knowing the whereabouts or habits of the user, or in provoking the user some kind of damage, such as depleting her monthly data plan or systematically consuming the smartphone battery for the rest of the day. Adversaries of this kind may have strong personal reasons to attack the user and, at the same time, may already have some information about their target that they could use, such as the user cellular network operator, phone number, e-mail address, phone model (*e.g.,* Android, iPhone, BlackBerry), or coarse-grained geographical location (*e.g.,* a city, or a state).

Assumptions Given the above adversarial model, we assume the adversary is someone socially close enough to the victim user that they share a connection through a

social app or service that includes a near real-time messaging facility, such as Facebook Messenger, Google Hangouts, Skype, WhatsApp, Viber, and SnapChat. Our concept of social strength is therefore not necessarily measured in terms of real-life friendship but is much more relaxed. In fact, it is not uncommon for users to have several hundreds of online friends [1]. Considering how easy it is to establish relationships in some online services [3], an adversary may even create one or more fake identities to use for the attack, depending on the case. Another assumption is that the cellular network operator of the user is known, and that it allows unsolicited traffic to reach the mobile devices from the Internet or from the cellular network itself. According to recent statistics on over 180 UTMS carriers worldwide [18], this is true in more than 50% of the cases. Given that the popularity of cellular network operators in various countries is typically very skewed, knowing the cellular network operator of the victim user does not represent an issue. Moreover, getting this type of information from an online friend may not be hard, considering that, for instance, Facebook has a mobile phone number field in the contact description. Finally, we do not make strong assumptions about the amount of bandwidth resources available to the adversary, although there is a relation between bandwidth and detection time, as we discuss in Section 4.3.

3 Delay-Based Network Activity Detection in Cellular Networks

The cornerstone of our user IP detection methodology in cellular networks is a method that leverages network delays to accurately infer the transmission patterns of a mobile device, such as a smartphone. In this section we describe the characteristics of cellular network radio resource assignment that enable it.

3.1 Radio Resource Assignment in Cellular Networks

Radio resources in cellular networks are allocated to mobile devices in relation to the volume of data they are sending or receiving from the network. This process is regulated by means of transitions in a Radio Resource Control (RRC) state machine that is associated to each device [20]. In 3G networks, these states are typically three: `IDLE`, `CELL_FACH`, and `CELL_DCH`, corresponding to no, low, or full radio resources allocated, respectively. Transitions from lower to higher resources states are triggered by some network activity, and are referred to as a promotions. The opposite transitions are instead referred to as a demotions. Although state transitions parameters can be independently defined by each mobile network operator [17,18], two general rules always apply. First, a promotion from `IDLE` is triggered when *any* amount of data has to be transferred, whereas a `CELL_FACH` to `CELL_DCH` promotion is triggered when the data rate exceeds a given threshold defined by the operator (*e.g.,* 2Kbit/s). Second, state demotions are triggered from the `CELL_FACH` and `CELL_DCH` state after a period of no network activity referred to as *tail time*. Tail times relative to the `CELL_FACH` and `CELL_DCH` states are operator-defined, although the former is typically longer than the latter as it consumes less radio and energy resources. As an example, we show the state machine configuration of a popular network carrier in Figure 1. We can notice how in this configuration a transition to the `CELL_FACH` state is first required in order

to get the full-resources available in the CELL_DCH state. Other operators may use a more aggressive configuration, allowing a direct promotion from the IDLE directly to the CELL_DCH state.

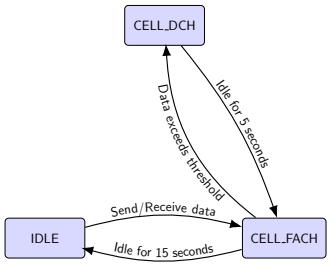


Fig. 1: Example of RRC state machine.

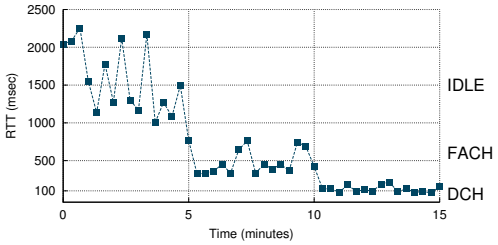


Fig. 2: Example of effect of network usage on observed RTT (round-trip time) towards the device.

3.2 Inferring RRC States From Network Delay Measurements

The device current RRC state and its responsiveness to network events are tightly related. This comes for two reasons. First, state promotions, especially those from the IDLE state, are time expensive, as they require a number of control messages to be exchanged between the device and the Radio Network Controller (RNC). Second, a device in the CELL_FACH state typically observes higher delays due to the lower amount of allocated radio resources and to its lower transmission power. A key observation is that the extra delays caused by promotions from the IDLE state and by the low resources of the CELL_FACH state are significantly higher than typical network delays, and can be easily distinguished from each other. To give an example, in Figure 2 we show a sequence of round-trip time (RTT) measurements performed every 17 seconds towards a device using the RRC machine state configuration shown in Figure 1. When the device is idle, the RTTs fall in the $[1\text{s}, 2.5\text{s})$ range. Such high delays are not produced at the network level, but are caused by promotions from the IDLE to the CELL_FACH state. Indeed, RTT measurements are spaced by an amount of time larger than the CELL_FACH tail time, which is large enough to make the state machine transit back to the IDLE state between measurements. After 5 minutes, a concurrent traffic on the device is generated with a rate of 0.5 kbit/s, enough to keep its RRC state machine in the CELL_FACH state. In this state, the RTTs drop into the $[250\text{ms}, 1\text{s})$ range, although they are still higher than expected, due to the low resources associated to the device. Finally, when the device is allocated full radio resources in the CELL_DCH state, the RTTs fall in the $[0\text{ms}, 250\text{ms})$, which is the actual network round-trip time between the measuring host and the mobile device. Overall, the strong difference between delays imposed at the various states makes a single round-trip time measurement a surprisingly robust and effective way to remotely infer the recent network activity of any given device in a cellular network.

4 User IPs Identification in Cellular Networks

In this section we present a novel methodology that leverages the delay-based RRC state inference to spot the IP of a target user in the cellular network. With this method, an adversary who has some *indirect* way to produce traffic on the target device can ensure it to be in a high-power state (*i.e.,* `CELL_FACH`, `CELL_DCH`) at specific moments, producing a distinctive network delay pattern on the mobile device. At the same time, the adversary looks for similar patterns across all the devices of the cellular network operator. This results in a set of candidate IPs which can be iteratively reduced in size by repeating the same procedure. The detection methodology, detailed in Algorithm 1, works in rounds. At each round, the `generate_traffic` function is used to generate

Algorithm 1 Pseudocode of the IP identification method

```
1: INPUT: IP_Range, nrounds, T_wait
2: for i := 1 to nrounds do
3:    RTTs := new map()
4:    generate_traffic_start()
5:    for all IP ∈ IP_Range do
6:       RTTs[IP] := measure_RTT()
7:    end for
8:    generate_traffic_stop()
9:    for all IP ∈ IP_Range do
10:      if is_match(RTTs[IP]) = False then
11:         IP_Range := IP_Range \ {IP}
12:      end if
13:   end for
14:   sleep(T_wait)
15: end for
16: return IP_range
```

traffic on the target device by sending messages to the its user through an instant messaging app. In the meanwhile, the current RRC state of all the devices in the `IP_Range` set is identified by measuring their round-trip times, as explained in Section 3. Round-trip times can be performed through ICMP echo requests (pings), or by sending SYN packets to a closed port and waiting for the relative RST packet. At the end of the measurements, traffic generation is paused and the `is_match` function is used to filter-out the set of all the devices whose radio was not at a high power state. Aside from the target device, this includes all the other devices in the network that were using the radio during the measurement. What enables this methodology to filter them out is the fact that mobile devices, as opposite to laptops or mobile hotspots, are not likely to constantly use the network resources for long periods of time, whereas the target device can be forced to transmit at will.

Methodology Parameters Our detection methodology takes three parameters, namely `IP_Range`, `nrounds`, and $\mathtt{T_{wait}}$. The first one, `IP_Range`, is the initial candidate set

of IPs assigned to the target user device. In case the operator assigns public IPs to its devices, a simple whois query with the operator AS name or number(s) reveals the initial IP set[1]. In case private IPs are used, the set of potential IPs may be very large, such as the private 10.0.0.0/8 subnet which is approximately 16M IPs wide. Starting from the whole subnet does not constitute an obstacle, but it still may slow down the detection procedure, requiring more rounds to shrink the candidate set of IPs to reasonable values. The initial `IP_Range` set can be considerably restricted if the user's coarse-grained geographical location is known. In fact, although the correspondence between IP address and location in cellular networks is not as strong as in wired networks [25], several other network features can be found (*e.g.,* the minimum round-trip time, and the RRC state machine configuration parameters) to effectively map IP addresses of mobile devices to a geographic area, like a big city or a state [18]. This would not require a large amount of bandwidth, and would not violate our attacker model.

The `nrounds` parameter determines the number of refinements the procedure can perform. A larger number of rounds helps reducing the candidate set, but, at the same time, requires a higher number of messages to be sent to the target user. Ideally, the adversary should use the smallest possible number of rounds, depending on his objective. For instance, to perform a DoS to the user device, the adversary may be satisfied to save the bandwidth needed for the attack by reducing the `IP_Range` to a few thousand devices, in line with our weak attacker model. For other tasks, the final `IP_Range` should be smaller. In any case, in Section 5 we show how just 10 rounds are sufficient to reduce the final `IP_Range` set to a handful of devices.

Lastly, the $\mathtt{T_{wait}}$ parameter determines the frequency with which instant messages are sent to the target user. In general, the higher the frequency, the less time it takes for the adversary to individuate the user IP. However, a very high message frequency may make the user suspicious about the adversary's intent, and may actually require more rounds (messages) to detect the target device IP (more on this in Section 5). In this case, a stronger social relationship with the user may help disguising the instant messages as a regular chat.

4.1 Indirectly Generating Traffic on the Target User Device

Our IP identification method requires the adversary to ensure that, in the time span the round-trip time to all the mobile devices is measured, the target user device radio is in a high power state (*e.g.,* `CELL_FACH` or `CELL_DCH`), yielding a distinctive sequence of low round-trip times. According to our attacker model (*cf.,* Section 2), the adversary is able to send the victim an instant message through one or more social messaging apps. These apps typically include a push-notification facility to notify users of incoming messages in a timely fashion, representing an ideal way to indirectly generate traffic on the victim user device at specific moments. To confirm this intuition, we experimentally tested the responsiveness of the push notification systems used by the most popular messaging apps available on the market today, namely, WhatsApp, Viber, Google Hangouts, Skype, Facebook Messenger and SnapChat. More specifically, we measured the time span between the instant the adversary sends a message to the victim

[1] `https://www.team-cymru.org/Services/ip-to-asn.html`

and the instant the victim device actually receives it, forcing the RRC state to be either CELL_FACH or CELL_DCH, depending on the configuration (*cf.,* Section 3). According to our measurements, push notifications of all the aforementioned apps alert the user always within a timespan of at most 5 seconds after the message has been sent, depending also on the initial RRC state of the device. Push notifications delays are therefore low enough for the adversary to control the observed delays towards the victim device in a relatively fine-grained way and with good accuracy. It is worth noticing that we did not observe any difference in responsiveness between messages sent when the device is active (*i.e.,* the screen is on and unlocked) or in sleep mode (*i.e.,* the device has not been recently used, and the screen is off). This allows an adversary to generate traffic on the device even without the user being immediately aware of it (*e.g.,* when the phone is muted, or at night). As an example, we show in Figure 3 the sequence of

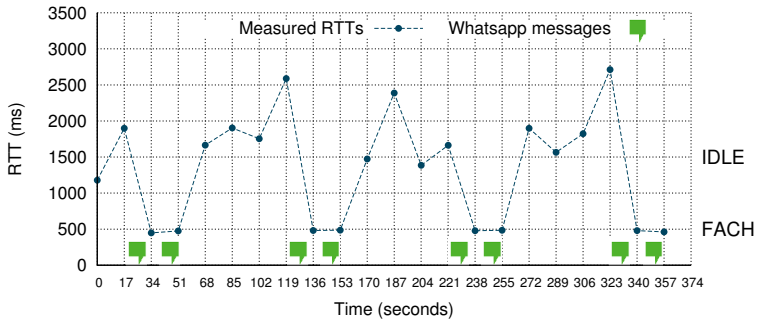


Fig. 3: Example of traffic generation on the target device by means of WhatsApp messages. To detect the CELL_FACH state, round-trip times are measured every 17 seconds.

radio states inferred while generating traffic towards a test device through WhatsApp messages. Round-trip time measurements are spaced by 17 seconds, which are slightly higher than the tail time of the CELL_FACH state in Figure 1. The figure shows how the state inferred 5 seconds after a message was sent is always CELL_FACH (*i.e.,* RTT $\in [300ms, 1s)$). We can also observe how the state machine does not transit back to the IDLE state when another instant message is sent before the CELL_FACH tail time expires. This is an important detail, as it enables an adversary to extend the time the target device radio is at a high power at will, if needed. Finally, observe how the traffic generated by the push notifications may not be high enough to trigger a transition to the CELL_DCH state. The adversary should therefore conservatively assume that the target device is in the CELL_FACH state after a message is received, even though concurrent traffic on the user device may actually make the radio transit at a CELL_DCH state. This does not represent a limitation though, as the difference between round-trip times at CELL_FACH and at IDLE is high enough to reliably tell the two states apart.

Overall, this experiment shows that a single round-trip time measurement can easily detect the network activity triggered by an instant message. We argue that, aside from those we experimentally tested, most of the instant messaging apps available on the

market today can be leveraged, given their real-time nature. In fact, according to our experience, any messaging app using the Google Cloud Messaging (GCM) push notifications is a good candidate. This is, for instance, the case of not just Google Hangouts, but also of Skype and Facebook Messenger. In principle, other apps may be used too, such as Facebook, Twitter, or GMail. However, having no strict real time requirements, their notifications may be delayed by several seconds, or even minutes, making it harder to detect the user IP.

Setting Up the Attack As a preliminary step, the adversary should compare the time it takes to perform a single round-trip time measurement towards the devices in the `IP_Range` set with the amount of time the traffic generated by the push notifications can keep the target user device radio at a high power state. Depending on the RRC state machine configuration, the latter may be the `CELL_FACH` tail time only, or the sum of `CELL_FACH` and `CELL_DCH` tail times. If this time is too short, the adversary may need to extend it by sending a sequence of well-spaced instant messages, as we have shown in Figure 3. To learn the RRC state machine configuration, the adversary can use the technique described by Qian *et al.* [17] either remotely or from a device under his control in the cellular network, as we did for our experiments. If the approximate geographical area where the victim device resides (*i.e.,* a city, or a state) is not known, the adversary may need to account for different possible `CELL_FACH` and `CELL_DCH` tail times. To avoid false negatives, the adversary should conservatively assume that the shortest tail times are used on the target device. This makes our detection methodology adaptive to any RRC state machine configuration an operator might use in his network.

4.2 Factors Affecting the Detection Accuracy

For our detection methodology to succeed in real-life scenarios, packet loss and delay variations caused by cross traffic or wireless signal interferences have to be taken into account. In this section, we quantify the impact these factors have on the IP detection accuracy.

Cross-Traffic Cross traffic represents a potential issue, as it is well known that, due to the use of large buffers, it might introduce extra network delay variations. However, the relative increase in round-trip time produced by cross traffic is still small with respect to the much higher delays observed when the device is in the `IDLE` state (*i.e.,* 500ms against more than 1 second). Only when cross traffic is close to the bottleneck link capacity for long time, the extra delay can reach the order of seconds [9] and the adversary could mistakenly infer that the RRC state was `IDLE`. Given the bursty nature of traffic in mobile networks [19] we deem this as an extreme scenario. In fact, mobile devices are typically used for short periods of time, and non-user generated traffic is produced by background apps and services regularly downloading short updates from the network (*e.g.,* social network status updates, emails, and so on). Moreover, even streaming apps, which are considered to be resource-hungry services, use temporary buffers to store up to several seconds of pre-fetched media content. To experimentally confirm the low impact of cross traffic on our detection procedure, we measured the round-trip

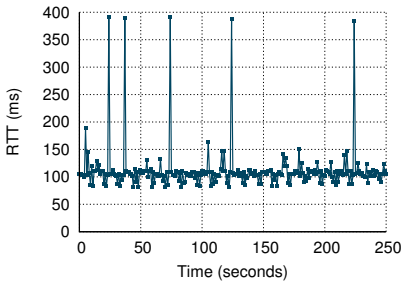


Fig. 4: Round-trip time measurements in presence of cross-traffic produced by a Youtube video streaming.

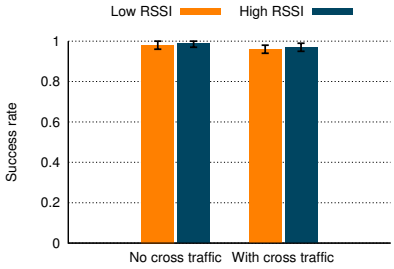


Fig. 5: Probability of measuring a low RTT ($\leq$ 1s) after a push notification under different signal strength and cross traffic conditions.

time towards a device under our control that was downloading a Youtube video. This well represents one of the most intensive network activities that can be triggered by a user. Results, presented in Figure 4, show that, despite the video being downloaded, for most of the time the measured round-trip times match very closely those typical of the `CELL_DCH` state, like those shown in Figure 2. As expected, only a few measurements show the effect of some queueing delay (*e.g.,* 400ms), but never reach values close to those typical of the `IDLE` state.

Wireless Signal Interferences Due to the loss-recovery mechanisms typically used in cellular networks, wireless signal interferences can introduce extra delays too. Garcia *et al.* [4] have experimentally shown that, in exceptional cases, this can introduce spikes of delay of up to 400 milliseconds in UTMS networks. These delays are still lower than the round-trip times generated at the `IDLE` state, thus not producing false negatives.

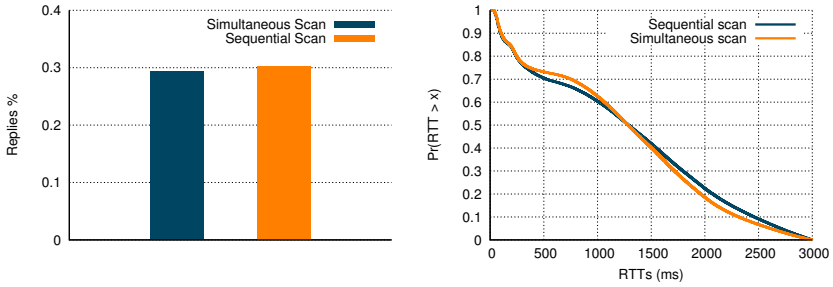
Network Packet Loss Since we can exclude wireless interferences as a direct cause of packet loss, in typical scenarios we do not expect packet loss to be very high or to last for long periods of time. To account for sporadic packet losses, if a round-trip time measurement fails, the adversary may conservatively assume the device is in `CELL_FACH` and keep it in the `IP_Range` set for an extra round.

Experimental Validation We experimentally evaluated the resilience of our detection mechanism to all the above factors with a device under our control. In our experiments, we simulated both the traffic generation and detection method by first sending a WhatsApp message to the target device, and then inferring the RRC radio state after 5 seconds the message was sent in order to account for the time it may take for the push notification to reach the device. During each test, the device was put under different cross-traffic and signal strength conditions. More specifically, cross traffic was produced by downloading a Youtube video, as in Figure 4. For what concerns the signal strength, we tested two different scenarios: One with poor and one with good signal,

corresponding to RSSI ~ -95dBm and RSSI ~ -79dBm, respectively. Each combination of cross-traffic and signal strength was tested 300 times. According to our results, shown in Figure 5, measured round-trip times are within the expected threshold in both cross traffic and poor signal strength conditions for more than 99% of the cases. Given these results, allowing a small percentage of round-trip times measurement to fail would make our detection mechanism resistant to a wide range of conditions.

4.3 Scanning the Mobile Operator's Network

To reduce the number of messages required to be sent at each round, the adversary should minimize the time needed to measure the round-trip times towards the devices in the `IP_Range`. For this task, a tool like ZMap could be used. ZMap is a network scanner by Durumeric *et al.* [6] that is specifically tailored for performing fast, large-scale network scans with commodity hardware. As a proof-of-concept we developed an ICMP echo request (ping) scanner module for ZMap and tested it against the 1.7M IPs wide address space of a popular cellular network that assigns public, reachable IPs to its devices. Taking as a reference the 15 seconds `CELL_FACH` tail time in Figure 1, and the 5 second to be waited before the messages arrive to the device, to perform a single round-trip time measurement on all the IP space in time would require 46Mbit/s using ICMP echo packets of 36 bytes. In practice, we observed that the number of IPs that are active at any given time in the address space is much smaller, that is, around 500K (30% of the total), which is consistent to what observed by Qian *et al.* in a U.S. mobile network carrier public address space [18]. This allows to restrict the initial `IP_Range` and reduce the bandwidth needed to just 14Mbit/s.



(a) Percentage of replies.

(b) CCDF of the round-trip times.

Fig. 6: Results with different scan strategies.

We experimentally verified whether large network scans produce some bottleneck in the cellular network infrastructure, which could have an impact on the detection accuracy. To do so, we compared the results obtained by scanning the whole public IP space of the operator in two different ways. First, with a simultaneous scan over the whole address space range. Then, by dividing the address space in smaller chunks (*i.e.,* 65K devices) and scanning each of them separately, waiting for 30 seconds between

subsequent scans. The results obtained are shown in Figure 6. In Figure 6a, we can observe how the number of replies received is in both cases very close to 30%, showing that no extra packets loss was introduced during the simultaneous scan. Figure 6b, comparing the two round-trip time distributions obtained, shows that no relevant extra queueing delay is produced either.

In Figure 7 we study the relation between size of the `IP_Range` set to be scanned, the bandwidth available to the adversary, and the number of messages to be sent at each round to keep the radio of the target device at a high power state for enough time, according to the state machine configuration of our test operator (shown in Figure 1). We can observe how, even with lower bandwidths (*i.e.,* 5Mbit/s), just one message buys the adversary enough time to scan a whole /16 subnet ($\sim$ 65K IPs). As we are going to show next, the exponential drop in size of `IP_Range` allows to use just one message per round in the majority of the cases.

5 Experimental Results

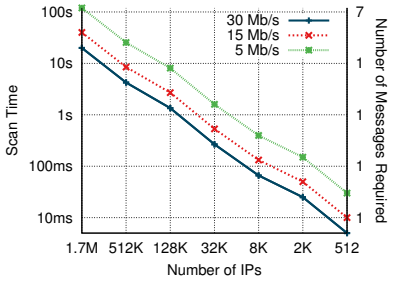


Fig. 7: Time required to send ICMP echo requests to all the IPs in `IP_Range` from our probing host.

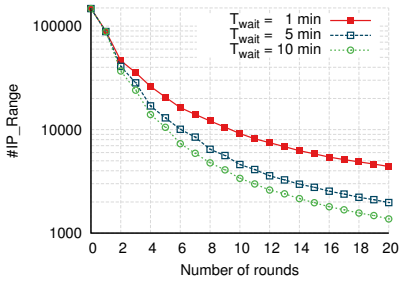


Fig. 8: Number of IPs left after each round of Algorithm 1 using different T_{wait} values.

We experimentally evaluated our IP detection methodology on a popular cellular network operator which assigns public, reachable IPs to its devices. As the target device, we used a Samsung Galaxy S Plus. As initial `IP_Range` set, we used four $/16$ subnets (*i.e.,* 262K IPs) from which IP addresses are typically assigned to the devices in our geographical region. The RRC state machine configuration used by the operator in the area is the one shown in Figure 1. Consistently with our experiments in Section 4.1, we conservatively instrumented the `is_match` function to keep in the `IP_Range` set all the devices that are either in `CELL_FACH` or `CELL_DCH` state (*i.e.,* RTT $\leq$ 1s), as our push notifications were not necessarily able to trigger a transition to `CELL_DCH` on the target device. Having fixed `IP_Range` and `nrounds`, the only parameter left to be chosen is T_{wait}, which determines the amount of time the adversary waits before starting a new round. We already discussed the pros and cons of using a low or high T_{wait} in Section 4, but during our experiments we observed that this parameter has also

a strong impact on the number of rounds required to reduce the IP_Range set: The lower the T_{wait}, the less IP_Range is reduced in size at each round. This effect can be explained by considering that network activity on mobile devices is typically produced in bursts (*e.g.,* when the device is being used, or when some background service downloads an update). Thus, the closer two rounds are in time, the higher is the probability that the devices that were transmitting (*i.e.,* not in IDLE) at the previous round will be still transmitting during the following one, and will not be removed from the IP_Range set. Using a larger T_{wait}, instead, increases the probability that the devices in IP_Range go back to the IDLE state between rounds. Starting from this observation, to test our detection methodology we used three different values of T_{wait}, namely 1, 5, and 10 minutes. We want to stress here that these values are just representative of what an adversary could use, and that the T_{wait} parameter does not necessarily need to be fixed. For instance, the adversary may very well disguise the messages sent to the victim user as a regular chat. In this case, the timings between messages (*i.e.,* rounds) to the victim could vary according to the conversation.

We performed 5 independent tests for each of the T_{wait} parameters in the timespan of three days, between 10am to 8pm. The measurement machine was hosted in our university's network, and we used the custom ping-probe module for ZMap. Our machine, an Intel Core2Duo with a 100Mbit network interface, takes around 10 seconds to perform round-trip time measurement when the IP_Range size is maximum (*i.e.,* 262K IPs), which corresponds to a rate of around 27 Mbit/s. For the instant messages we used WhatsApp, as it provides a handy API through which automated tests can be performed. A number of other apps may be used as well, as discussed in Section 4.1. In order to account for packet losses, if a device stops replying for at most two rounds, we assume it was in CELL_FACH and we don't remove it from the IP_Range set. Finally, we used 1 second as a round-trip time threshold to detect the IDLE state, consistently with the experiments presented in Section 4.1.

Results in Figure 8 show the number of remaining candidates in IP_Range after each round, averaged across all the tests. First of all, we can observe how, during the first round, the size of IP_Range suddenly drops from 262K to just around 80K. This is caused by IPs not always being active, as we already observed in Section 4.3. As we anticipated, when the T_{wait} parameter is too low (*i.e.,* 1 minute), the size of the IP_Range decreases very slowly with respect to the other two cases, which, instead, show similar performances. Using wait time larger than 10 minutes may further improve the results, but would also be more time expensive for the adversary. Surprisingly, with T_{wait} set to 5 minutes and 10 minutes, in just 10 rounds the IP_Range gets reduced to between 2K and 3K IPs, that is, 7% and 3% of the initial size of the IP_Range set. This result may be good enough if the objective of the adversary is to save the resources needed to perform a DoS attack against the victim user. For other type of attacks, the adversary may want to get an even smaller set of candidate IPs of the victim device. As we show next, this can be achieved with just a slightly more accurate model of the Internet usage pattern of the victim.

5.1 Monitoring the Network Usage

One characteristic we observed about the set of devices that stay in the `IP_Range` for more than 10 rounds across all our experiments, is that they are mostly characterised by periods of network activity that are exceptionally long with respect to those typical of mobile devices [10,18]. To get a more clear view of this phenomenon, we randomly selected 10K IP addresses that resulted to be online and measured their round-trip time every minute for one hour. Figure 9 shows that around 80% of the devices were found

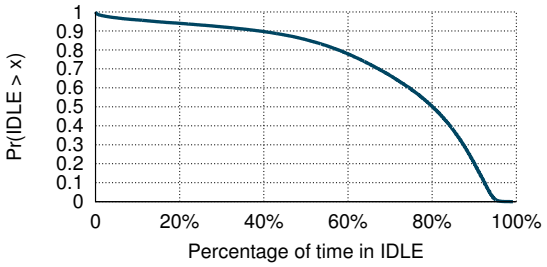


Fig. 9: CCDF of the percentage of time the devices were found in the IDLE state.

to be in `IDLE` for more than 60% of the time. This idle time should be consistent to the typical traffic patterns of mobile devices [10,18], who are also very limited in terms of energy autonomy [16]. Following this intuition, we believe that typical mobile devices, such as smartphones or tablets, are unlikely to exhibit traffic patterns with very long periods of network activity. In fact, it is reasonable to expect that the cellular network operator uses its IP address space for other type of services too (*e.g.,* 3G hotspots, publicly accessible WiFis, 3G USB sticks, and so on). For this reason, the adversary may safely assume that, in the long run, the target IP will be found to be idle for at least a P_{idle} percentage of time, and remove from the `IP_Range` set the devices with a higher percentage of transmission time. This could computed by keep measuring the round-trip time of the devices even in the time period between consecutive rounds, that is, when no message is sent to the user with `generate_traffic()`. After a sufficient number of rounds, an accurate profile of the network usage over time of the devices in `IP_Range` can be built and used to filter out devices with an unexpectedly high network activity. We tested the effectiveness of this profiling technique in our experiments by monitoring the state of the devices in `IP_Range` every minute and using different values of P_{idle}. The results, presented in Figure 10, show that, after 10 rounds, even conservatively assuming an at most 40% of idle time, we can remove up to 87% of the devices when T_{wait} set to both 5 and 10 minutes, corresponding to 50 minutes and 1 hour and 40 minutes, respectively. If the adversary can make a stronger assumption on the idle time of the target device, such as by setting P_{idle} to be 80%, then the final `IP_Range` set gets reduced to just 6 and 3 devices respectively. Using a larger number of rounds helps further decreasing the possible number of user IPs with a less restrictive assumption on their idle time. For instance, with 15 rounds, corresponding to 1 hour and 15 minutes

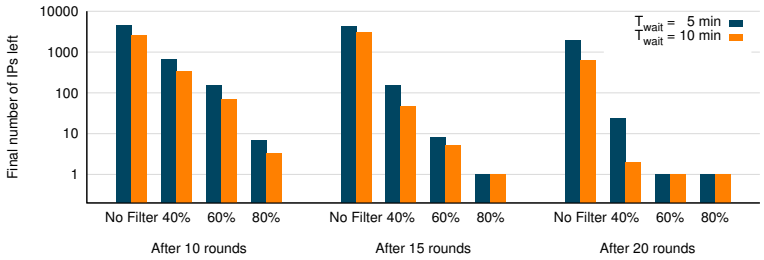


Fig. 10: Filtering IPs based on the percentage of idle time

and 2 hours and 30 minutes, assuming P_{idle} to be 60%, the number of final IPs is less than 10. We exactly identified the target device IP using 15 rounds and $P_{idle} = 80\%$ or with 20 rounds and $P_{idle} \geq 60\%$. Although we leave a thorough analysis of the typical activity patterns of smartphones as a future work, our results show that this filtering technique can achieve a very high accuracy with just 10 or 15 instant messages to the victim user. Moreover, our results show that using a T_{wait} time of 10 minutes does not provide a significant advantage. So, if only a limited amount of time is available for the adversary to restrict the user IP, T_{wait} may be set to 5 minutes.

In general, depending on what additional information is available about the victim, the adversary will always choose a time window that maximizes P_{idle} of the target device during the attack. In fact, there are several scenarios in which the interaction between a user and the mobile device may be minimum: while driving (maybe during commute), during lectures (if the target is a student or professor), during trials (if the target is a judge or lawyer), at night, *etc.*.

5.2 Fingerprinting the Mobile OS

Perhaps surprisingly, the operating system is another information that can be inferred while measuring the round-trip times towards the devices in `IP_Range`, and that the adversary can exploit for an easier individuation of the target user IP. This is made possible by the choice that different operating systems make of the initial time-to-live (TTL) values to the ICMP echo reply they generate. For instance, the TTL distribution obtained during a sample scan, reported in Figure 11, is characterised by three steps in the vicinity of 64, 128 and 255, which correspond to the initial TTLs used by Linux (including Android) and Apple OSes (TTL 64), Windows (TTL 128), and BlackBerry (TTL 255) devices.[2] A similar TTL-based fingerprinting technique has been described by Vanaubel *et al.* [24]. Interestingly, although they use the same initial TTL, iOS devices can be told apart from Linux and OSX ones by observing that iOS listens for external connections on the TCP port 62078, which is reported as `iphone-sync` by nmap. Thus, using ZMap to perform a parallel SYN scan on this port the adversary can identify Apple mobile devices with just the same amount of effort and time required by a round-trip time measurement towards the devices in the `IP_Range` set.

[2] We could only use a limited number of BlackBerry phones to confirm their initial TTL value.

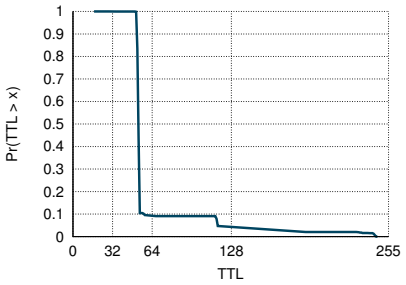


Fig. 11: ICMP echo reply TTL distribution during a sample scan.

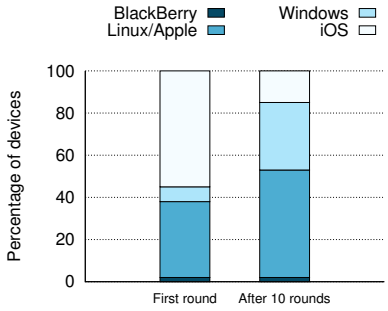


Fig. 12: OSes inferred before (left) and after (right) Algorithm 1 is run ($\texttt{nrounds} = 10$, $\mathrm{T}_{\texttt{wait}} = 5\mathrm{m}$).

Overall, if the operating system of the victim user's device is known to the adversary, this mix of TTL-based and port-based OS detection methods allows to easily restrict the initial `IP_Range` set, reducing the number of rounds required to find the IP of the target device. As an example, in Figure 12 (left column) we show the distribution of the inferred OSes on the initial `IP_Range` we used for our experiments. We can observe how the low percentage of Windows and BlackBerry devices makes their users particularly vulnerable to our detection method. For instance, the IP of a BlackBerry user can be identified with just 15 rounds and P_{idle} of just 40%. To conclude, notice that, as the number of rounds increases, the OSes distribution changes considerably (compare left and right column of Figure 12). In particular, the percentage of Windows devices after 10 rounds increases more than 4 times, whereas the percentage of iOS devices decreases. Considering that devices that stay in the `IP_Range` set for several rounds are those with the lowest idle time (*cf.,* Section 5.1), we deem this as a confirmation of the intuition that these devices may not be actual smartphones, but, rather, other devices getting connectivity from the cellular network.

6 Discussion

In this section we investigate the feasibility of our detection methodology, and discuss alternatives and possible countermeasures.

6.1 Firewalls and NATs

A requirement of our detection methodology is the possibility of directly reaching the target mobile device in the cellular network. Thus, NATs or firewalls in the path between the adversary measurement host and the device may constitute an issue. In cellular networks, these mechanisms are typically deployed at the edge of the Radio Access Network (RAN), immediately before the public Internet. According to our experience, further confirmed by recent findings [18], todays' cellular networks can show a wide

	Africa	Asia	Europe	Oceania	N. America	S. America
# Carriers	5	33	35	3	9	14
Subscribers %	7	56	27	40	54	73

Table 1: Number of mobile network carriers whose mobile devices are assigned public, reachable IP addresses. Subscribers percentage is a coarse estimation based on publicly available data.

variety of configurations, which differ on whether firewalls blocking unsolicited traffic towards the mobile devices are deployed, or whether NAT is used (or both). In the former case, the detection methodology would not work, as the packets needed to measure the round-trip times would be blocked. In the latter case, the only potential obstacle is NAT. However, we observed that when mobile devices are assigned an address in the private IP space, they are still freely reachable by other devices in the same cellular network, allowing NAT to be circumvented by an adversary that has access to a device in the target cellular network. Considering that subscribing to a mobile data plan is relatively cheap, we believe this is not a strong requirement, even for reasonably weak adversaries.

Finding Public IP Operators To get a more complete view of the cellular network operators worldwide that allow their devices to be directly reached from the open Internet, we independently collected a list of mobile network operators that assign public, reachable IP addresses to their devices. As opposed to previous approaches [18], we used a centralised approach. As in Section 5.2, we leveraged the fact that iOS devices listen for external connections on `iphone-sync` port. This allowed us to spot all the publicly reachable iOS devices by scanning the entire Internet IP space for hosts listening on this port using ZMap. The scan took 10 hours, and yielded $\sim$ 9.4M IPs in 6315 unique Autonomous Systems (AS) distributed across 189 countries. We were able to associate $\sim$ 7.5M of these devices ($\sim 83\%$ of all the hosts we detected) to 103 among the main network operators in each country [3], by looking at the name of their originating ASes. Table 1 gives, the number of operators found in each continent, and their total estimated share of customers. Given the strong market penetration of iOS devices, we believe this is a fairly accurate preliminary list of mobile operators that provide unrestricted access to their devices from the open Internet. Our results show that the potential number of users that can be identified with our methodology is far from negligible. In fact, in all the continents, the mobile carriers allowing a direct access to their devices own a substantial part of the market share (in terms of number of subscribers). Overall, our evaluation provides a much broader view of this phenomenon with respect to previous studies, with a significantly smaller effort. This comes at a price of a lower accuracy. For instance, associating an AS to its corresponding mobile carrier is not always trivial, and may require a manual examination of the AS names.

[3] `http://en.wikipedia.org/wiki/List_of_mobile_network_operators`

6.2 IP Duration

The amount of time a mobile device is assigned a given IP address in the cellular network is another aspect that may affect our detection methodology. The longer the time, the easier will be for the adversary to spot the device IP, and greater it will be the possibility to harm the end-user. Typically, the IP address assigned to a mobile device never changes as long as the device gets not disconnected from the network. This holds for both cellular networks that assign addresses in the public IP space, and for operators that, instead, use NAT. In the latter case, the public IP assigned to connections originating from the device may change in an unpredictable fashion, depending on how the network is configured, as also reported by Balakrishnan *et al.* [2]. This does not constitute an issue, though, because, when NAT is used, the adversary would need to get access to the private side of the cellular network, where IP typically change only when a disconnection occurs. For these reasons, the habits of mobile users play an important role in our attack scenario. For instance, if the user switches to a WiFi network, or turns the device off, then the device IP will almost surely change the next time a data connection to the cellular network is established. However, even for users that have access to WiFi during the day, there are still large time windows across the day in which their mobile devices are connected to the mobile network, such as when the user is on the move, or in some public place (*e.g.,* a pub), and so on. This is confirmed by a mobile app dataset used by Qian *et al.* [18], according to which for more than 80% of the times, a mobile device is continuously connected to the mobile network for more than 4 hours. This is a time frame that leaves more than enough time for the adversary to both identify the user device with good accuracy and to perform a focused attack, like a resource drain attack [14].

6.3 Alternative Approaches

An alternative to our attack could be that of tricking the user to visit a malicious website under the adversary's control that keeps a record of the connecting IPs. While this would require less resources on the adversary's side, there are two main reasons to prefer our detection methodology. First, it does not assume the active participation of the victim user, which allows the adversary to detect the device IP even when the device is left unattended, as mentioned in Section 5.1. This would reduce the chances that the user takes any countermeasure on time, assuming that the push notifications triggered by the adversary raise any suspicion at all. This highlights the second advantage of our methodology, that is, the fact that it is harder to be detected as it leverages the (misleading?) perception that push notifications represent a safe channel that cannot be exploited by external adversaries. On the other hand, we believe that being repeatedly asked to follow a certain link is more likely to raise some suspicion. For this reason, our methodology is more suitable in cases where the adversary is interested in attacking the user multiple times, like in the location tracking scenario mentioned in Section 1.

6.4 Countermeasures

The most obvious countermeasure the operator can implement to block our IP detection methodology is to deploy a firewall that does not allow incoming unsolicited traffic to-

wards the devices inside the network. From the user's perspective, this may come at the cost of limiting the possibility of using P2P applications or of hosting publicly accessible services in the cellular network. However, this does not constitute a real issue as it is very uncommon to have such services hosted on mobile devices. From the operator's perspective, instead, completely blocking all incoming traffic makes it very difficult to troubleshoot faults or misconfigurations inside the network. For this reason, a very common practice is to allow at least ICMP messages in echo-reply mode. We believe that the most effective way would be to implement a firewall directly on the mobile devices themselves, without relying on the intervention of the operator. This way, all outgoing ICMP traffic and unsolicited TCP connections it could be safely blocked, preventing an adversary to remotely probe the device. A less obvious but more effective way for the operator to countermeasure this attack would be to deploy IPv6, with an addressing scheme robust to network scanning (*e.g.,* DHCPv6 with non sequential addresses). Given the huge address space of typical IPv6 subnets, this would make the scanning unfeasible even if we consider a much stronger adversarial model.

7 Related Work

Our mechanism for inferring RRC states from network delay measurements builds upon recent findings about the characteristics of cellular network resource allocation. Qian *et al.* [17] propose a way to fully characterise the RRC state machine by just externally probing a device. In [16] they improve on their previous characterisation methodology. Perala *et al.* [15] introduce a 3G Transition Triggering Tool (3G3T), to determine RRC state transition parameters used in 4 target cellular networks across 3 different countries.

A number of attacks on cellular networks have been presented so far. Traynor *et al.* [23] study how even relatively small botnets of mobile phones can degrade the network service in large-sized regions. Peng *et al.* [14] provide a detailed security evaluation of the mobile data accounting architecture. They also describe a stealth-spam attack that drains the mobile data plan of a target user. Lee *et al.* [12], show that it is possible to overload the radio network controller in a UMTS network by means of control messages. Qian *et al.* [18] present a way to create a map of IPs in a geographical area that can be used in a focused signaling DoS attack to the network.

Several attacks have been proposed that use network delay as a side-channel. Hopper *et al.* [8] show how round-trip times allow malicious web servers to link client requests traversing the same circuit in Tor. Ling *et al.* [13] investigate a network-delay based side-channel attack to infer web sites accessed by a user trough a VPN/SOCKS proxy. Gong *et al.* [7] show how the same information can be inferred from the round-trip time between a probing host and a home DSL router.

Stober *et al.* [22] show that an UMTS eavesdropper can identify a smartphones by means of the network traffic it generates. Le Blond *et al.* [11] show how inconspicuous Skype calls can be used to get coarse-grained users mobility over time. Their method allows to identify the public IP address of a mobile device, which does not correspond to the actual IP address if the user is behind NAT. On the opposite, our technique works also from within the private cellular network address space, exposing mobile also users

behind NAT. Moreover, our method is more general, as it can leverage any instant message application, Skype included.

8 Conclusions and Future Work

In this work, we presented a novel method that leverages the delay-based radio state inference in combination with real-time push notification subsystems to identify the IP of a target user in the cellular network. The information obtained in this way represents a potential threat to mobile users, as it permits even weak adversaries to perform focused attacks on users using a small amount of resources, such as inferring the user location, depleting the user data plan, or inferring the user Internet activity. Our results, performed on over 260K IPs of a large cellular network, show that, with just 10 messages, the potential set of IP addresses gets reduced to just 3%. With a more accurate model of the victim user Internet usage pattern, the target user device can be correctly identified with just 15 messages. A further improvement can be achieved assuming the OS running on the target device (*e.g.,* Android, iOS, Windows, or BlackBerry) is known. As a future work, we are investigating the possibility of using the radio state identification technique to deanonymize Tor [5] users in the cellular network. In addition, we plan to evaluate whether remotely monitoring the sequence radio states of a mobile device can be used to build an accurate fingerprint of the Internet traffic of its user. We believe our work may introduce a new, more general research direction on delay-based user fingerprinting on cellular networks, which we intend to explore in the near future.

References

1. Backstrom, L., Boldi, P., Rosa, M., Ugander, J., Vigna, S.: Four degrees of separation. In: WebSci. ACM (2012)
2. Balakrishnan, M., Mohomed, I., Ramasubramanian, V.: Where's that phone?: Geolocating IP addresses on 3G networks. In: IMC. ACM (2009)
3. Bilge, L., Strufe, T., Balzarotti, D., Kirda, E.: All your contacts are belong to us: automated identity theft attacks on social networks. In: WWW. ACM (2009)
4. Cano-Garcia, J.M., Gonzalez-Parada, E., Casilari, E.: Experimental analysis and characterization of packet delay in UMTS networks. In: NEW2AN. Springer (2006)
5. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. Tech. rep., DTIC Document (2004)
6. Durumeric, Z., Wustrow, E., Halderman, J.A.: ZMap: Fast Internet-wide scanning and its security applications. In: USENIX Security (2013)
7. Gong, X., Borisov, N., Kiyavash, N., Schear, N.: Website detection using remote traffic analysis. In: PETS. Springer (2012)
8. Hopper, N., Vasserman, E.Y., Chan-Tin, E.: How much anonymity does network latency leak? TISSEC 13(2), 13 (2010)
9. Jiang, H., Liu, Z., Wang, Y., Lee, K., Rhee, I.: Understanding bufferbloat in cellular networks. In: CellNet SIGCOMM Workshop. ACM (2012)
10. Jo, H.H., Karsai, M., Kertsz, J., Kaski, K.: Circadian pattern and burstiness in mobile phone communication. New Journal of Physics 14(1), 013055 (2012)

11. Le Blond, S., Zhang, C., Legout, A., Ross, K., Dabbous, W.: I know where you are and what you are sharing: exploiting P2P communications to invade users' privacy. In: IMC. ACM (2011)
12. Lee, P.P., Bu, T., Woo, T.: On the detection of signaling DoS attacks on 3G wireless networks. In: INFOCOM. IEEE (2007)
13. Ling, Z., Luo, J., Zhang, Y., Yang, M., Fu, X., Yu, W.: A novel network delay based side-channel attack: Modeling and defense. In: INFOCOM. IEEE (2012)
14. Peng, C., Li, C.y., Tu, G.h., Lu, S., Zhang, L.: Mobile data charging: new attacks and countermeasures. In: CCS. ACM (2012)
15. Perala, P.H., Barbuzzi, A., Boggia, G., Pentikousis, K.: Theory and practice of RRC state transitions in UMTS networks. In: GLOBECOM Workshops. IEEE (2009)
16. Qian, F., Wang, Z., Gerber, A., Mao, Z., Sen, S., Spatscheck, O.: Profiling resource usage for mobile applications: a cross-layer approach. In: MobiSys. ACM (2011)
17. Qian, F., Wang, Z., Gerber, A., Mao, Z.M., Sen, S., Spatscheck, O.: Characterizing radio resource allocation for 3G networks. In: IMC. ACM (2010)
18. Qian, Z., Wang, Z., Xu, Q., Mao, Z.M., Zhang, M., Wang, Y.M.: You can run, but you cant hide: Exposing network location for targeted DoS attacks in cellular networks. In: NDSS (2012)
19. Ricciato, F., Hasenleithner, E., Romirer-Maierhofer, P.: Traffic analysis at short time-scales: an empirical case study from a 3G cellular network. Transactions on Network and Service Management 5(1), 11–21 (2008)
20. Romero, J.P., Sallent, O., Agusti, R., Diaz-Guerra, M.A.: Radio resource management strategies in UMTS. John Wiley & Sons (2005)
21. Soroush, H., Sung, K., Learned-Miller, E., Levine, B.N., Liberatore, M.: Turning Off GPS is Not Enough: Cellular location leaks over the Internet. In: PETS. Springer (2013)
22. Stöber, T., Frank, M., Schmitt, J., Martinovic, I.: Who do you sync you are?: smartphone fingerprinting via application behaviour. In: WiSec. ACM (2013)
23. Traynor, P., Lin, M., Ongtang, M., Rao, V., Jaeger, T., McDaniel, P., La Porta, T.: On cellular botnets: measuring the impact of malicious devices on a cellular network core. In: CCS. ACM (2009)
24. Vanaubel, Y., Pansiot, J.J., Mérindol, P., Donnet, B.: Network fingerprinting: TTL-based router signatures. In: IMC. ACM (2013)
25. Xu, Q., Huang, J., Wang, Z., Qian, F., Gerber, A., Mao, Z.M.: Cellular data network infrastructure characterization and implication on mobile content placement. In: SIGMETRICS. ACM (2011)