# Security LoopHoles in Wireless Networks

Yuhui Feng, Ganga Reddy Tankasala, Jia Guo, Wei Tang

December 2015

## 1 Introduction

With the rapid increasing of smart devices in recent years, people rely much on cellular networks such as its data service and voice service. Hence, mobile data charging (MDC) is a crucial part of its operation, which requires specific standards of authentication, authorization, and accounting. Currently, for most operators, data were charged according to the volume that customers used. They believed, the user who was billed for the data is the one who used the data. However, the accounting system is not transparent for customers. It means that customers might pay for the data which they never used.

Several papers have been studied in MDC area. These findings demonstrate that MDC system is not as secure as expected. In fact, a few loopholes existed which can be utilized to get free data. Meanwhile, some mechanisms would caused customers pay for more data. Peng et al. [2] raised the insecure of MDC system by enabling "toll-free-data-access-attack" and "stealth-spam-attack" successfully. In addition, Peng et al.[4] found that charging error can be as large as more than 450MB for three hours. In recent work, Peng et al.[3] proposed that even simple attacks, for example "hit-but-no-touch-attack", can damage cellular networks.

Moreover, cellular networks itself has been progressively upgraded from the very beginning. The latest evolution, 4G LTE cellular networks, occurs in '10s and is quickly familiarized all over the world. However, it caused several traps when proposing new voice solutions. Tu et al.[8] demonstrated that data service is affected by the most popular CSFB-based voice service, such as throughput drop and 4G connectivity lose. Also, Tu et al.[5] analyzed root causes of these attacks and launched "ping-pong-attack" and "4G denial-of -service attack" to exhibit vulnerabilities. At the same time, Li et al.[1] pointed out that free data access can be gained, continuing data access can be shut down, an ongoing call can be subdued by exploiting loopholes.

To further understand the vulnerabilities of control-plane, Tu et al.[7] proposed that signaling diagnosis can span design defects and operational slips. Tu et al.[6] examined the interactions of control-plane protocol in mobile networks.

In this project, the root causes of three attacks are first analyzed. Second, the implementation process of these attacks is explained clearly. Finally, the verification of these attacks is given out.

## 2 TTL Attack

For current MDC system, the accounting operation is done at the core gateway (P-GW) once any data packets pass through. After that, data packets keep moving until reach their destination. The time-to-live (TTL) field of IP of data packets decides how far these packets can arrive. It is decremented each time the packet passes an intermediate router. Once it gets 0, this packet would be discard no matter where it is. Hence, if the value of TTL is not accurate or is malicious modified, these packets would never get to their planned destination.

TTL attack is a malicious attack which causes users being overcharged. In this attack, the value of TTL should be large enough to ensure the packet can pass the P-GW, which means this packet would be on the bill of the user. Meanwhile, the value of TTL should be small enough to arrive the user's device, which means the user would never see this packet. Given an improper TTL value as analyzed former, the user would be charged for the data which never arrive the user's device.

## 3 PingPong Attack

4G LTE(Long Term Evaluation) is the recent and latest development in cellular network technology to offer mobile and wireless access to smart devices including but not limited to phones and tablets. It is expected that number of LTE connections by 2017 would exceed 1 billion.

The LTE designed to improve data services ( 100 - 300 Mbps) over existing 3G/4G which are limited to (10 - 20 Mbps). While legacy systems supports CS(circuit switching) for voice traffic , PS(Packet Switching) for data services, LTE is designed to support PS only. However the PS only decision was to inherit the advantages of Internet technology aimed at providing higher bandwidths for exponentially growing mobile data demands.

However PS is not an ideal choice for voice traffic which has been the strictly important service for subscribers. To incorporate the voice traffic in LTE, two solutions are proposed namely CSFB(Circuit Switched Fallback) and VoLTE (Voice Over LTE). In the CSFB mechanism, the call requests is transferred from LTE to legacy 3G System forcing the mobile to 3G mode, once the call is completes,CSFB moves the device back to 4G mode. VoLTE levarages the existing Voice over Internet protocol (VoIP) like Skype, Hangouts etc... but with additional reservations for voice traffic in LTE to guarantee Quality of Service (QOS).

In this work we have covered one of the loopholes in the CSFB mechanism which is termed as "Ping-Pong" attack. The nature of the attack is to push the device from LTE- 3G and 3G-LTE. The oscillation of the device between two models like nature of ping-pang game and hence the name. The adverse affects of this attack are loss of throughput, battery drain, possible loss of network connection to external Internet services.

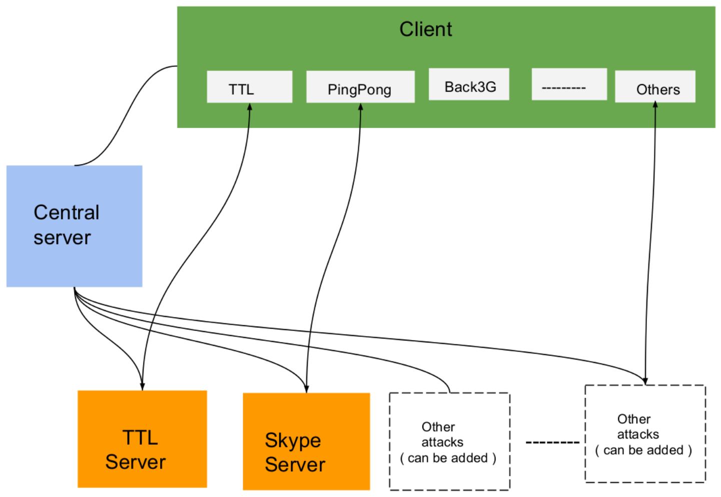


Figure 1: High Level overview of the framework

## 4 Back3G

The most popular voice solution of 4G LTE cellular networks is circuit switched fallback (CSFB). The main idea is to utilize 3G CS domain to serve CS-based voice calls. When a 4G user is called, the request routed to GMSC/MSC would requests MME to page. Once it is found, MME triggers 4G LTE networks to 3G networks and established the voice call once it is connects to 3G.

Back 3G attack is an attack which compels the user to connect 3G networks, which prohibits the user's connection of high-speed 4G access. Once a 4G user is called, the system would switch its cellular networks from 4G LTE back to 3G, which means suddenly the speed of network would drop down. If the calling time is short enough, the user even cannot realized this attack. The user would get the worse quality of network, such as opening a picture would take a long time, but the user can not figure it out.

## 5 Implementation

The attack framework we have developed consists of a mobile (client) and a central server which can be mounted behind any cloud. The Server caters to the attack requests from a client as part of reactive attack strategy compared to pro active attack strategy where the server keeps attacking a client without any request/consent from the client. The framework is designed to be of reactive strategy as our intention is to academically evaluate some of security loopholes in wireless networks. One of the main limitations to design an active attack framework is that there is a possibility of the applications being used ignorantly (to carry any attacks) and we do not want a reason behind it.

## 5.1 Server

The main goals taken in to consideration while designing centralized attack server framework are

Simplicity : Simplicity in understanding/analyse any attack mechanism of the server.

Flexibility : Flexible enough to add/modify/remove any attack from the server.

Efficient: Though not a strict requirement, it's always best to save as many CPU cycles as possible.

As shown in Figure 1, the attack framework has a central server which initiates the attack servers like (TTL Server, Skype Server) on request from client. The attack servers are designed to stop once the attack us complete instead of waiting for any further requests from client. Its the central Server which does the waiting and starts the attack servers on demand.

## 5.2 Client

In the client side, we build the application in Android devise, since it's easy to implement these attacks and it's free. Based on the main academic goal, we build a framework in devise side also, which will be easily to extended to implement other attacks and explore more loopholes. We build the functions and methods in our whole application object-oriented, some nice and easy APIs can be calling from the extended works.

### 5.2.1 CommunicationSocket

Communication Socket is a java class to handle the communication with server. It is essentially a integrated UDP socket and offers the functions of sending, receiving messages and flushing memories.

In the constructor, server's address and port number are needed to construct the instance.

$public\ CommunicationSocket\ (String\ serverAddr,\ int\ port)$

$sendPacket(String\ info)$ is offered to send UDP socket of the info to the server. $receivePacket(int\ timeout)$ and $receivePacket()$ are two methods to receive UDP packet and they will return a string of the content. The difference of them is that one provide a timeout for when receive a packet, but another would block the listening port until a packet is received.

$flush()$ method is to flush the receive buffer of the listening port, and it will return a Boolean variable to indicate the receive buffer is empty or not.

### 5.2.2 DataSet

DataSet is a java class to handle the time series data, which needs to plot in the UI. Basically, DataSet contains a vector of DataUnit, each of which stands for a set of values for a given time t. DataUnit and DataSet will be introduced in detailed in the following.

1. DataUnit

DataUnit is a class, which contains a time stamp. a vector a different data at each time, a vector of dataType to described each of the data.

Getter and Setter methods are implemented for each field, and the method of *getDataDimension*() is given to get the dimension of this DataUnit. Note, the data vector is associated with a datatype vector, they are corresponded to each other one by one. For example, {"3.0", "1.0"} for data, while {"Operator Data", "Local Data"} for dataType, all these means that operator data is 3.0MB, while local data is 1.0MB

2. DataSet

DataSet is a vector of DataUnit, which is to handle several time series like data. It offers *size*() method to return the size of the DataSet, getter and setter methods to operate the vector, and a *clear*() to clear the vector when needed.

### 5.2.3 GraphPainter

GraphPainter is a java class to handle the data virturalization and time series plotting. This plotting API can be reused anyone who wants to show their experiment results in real time. It provides the following interfaces:

1. Two kind of constructor

Constructor 1: it is the constructor for the state-like labels graph

$public\ GraphPainter(SurfaceView\ surface,\ Vector < String >\ labels,\ Vector < Vector < Paint >>\ paint)$

Argument 1 specifies a SurfaceView in layout to paint on.
Argument 2 specifies a Vector of labels for Y axis. For example {"2G", "3G", "LTE"} will make Y axis a network state indicator.
Argument 3 specifies different paints for different data groups. The inner Vector should be 3 dimensional, specifying the paint for data points, line segments linking data points and bars between data points and X Axis. And for simplicity, we can accept null in this argument and a default painter will the first two kind of data.

Constructor 2: it is the constructor for the exact-value data graph

$public\ GraphPainter(SurfaceView\ surface,\ String\ unit,\ Vector < Vector < Paint >>\ paint)$

This constructor will make the graph a data plot, Argument 2 specifies the unit of the data (e.g. "MB").

2. Plotting schedule

*public void schedule(DataSet dataSet, int interval)*

On calling this method, the Painter will start painting data every interval milliseconds on the surface. The data to paint is given by a DataSet, which is described in previous DataSet API. Here, For example, given DataSet={ DataUnit1 }, DataUnit= {data={1, 2}, datatype={"record 1", "record 2"} }, the painter will paint two points ($x_1$, 1) and ($x_1$, 2), representing the first entry of "record 1" and "record 2", respectively, where $x_1$ is the x Axis.

## 5.3 TTL Attack

### 5.3.1 Server Section

The server for TTL attack is developed to handle both attack and echo requests from clients on a pre-determined port. The clients sends a UDP request packet with two integer parameters namely ttl and size which cover the first four and next four bytes of the payload respectively. ttl refers to the desired TTL of the packet that server send and the size refers to the size of the attack requested in MegaBytes (MB).

Echo Request: If the parameter size is maintained zero in the request, then simply a UDP response packet with desired TTL is set and sent to the client. A series of echo tests will be implemented by the client to find out the highest TTL value which is sufficient to just reach the gateway of the core network.

Attack Request: If the parameter size is not zero in the request, then a series of UDP packets will be sent to the client until the entire total size of the packets sent exceed the requested size. We have restricted payload of each UDP packet to a fixed size of 1KB. If the size requested in 1, then 1000 ( as 1000 * 1 KB = 1 MB) UDP packets will be sent to the client. In order to limit the size of the attack , we have hard coded the max size of limit to 10 which means that any request with size greater than 10 MB will be limited to 10 MB.

### 5.3.2 Client Section

In TTL Attack, we offers two modes of attacks. The first one is default attack, which is to implement a valid 3MB attack. In this mode of attack, client first will do a ttl probing by using Echo Request to find the valid ttl to implement the attack. Then a valid a ttl with 3MB size of Attack Request are sent to the server, and a 3MB ttl attack will be launched by the server.

The second one is a custom attack that the user can define their own attack. When changed the switch to the non default mode, ttl value and attack size value can be set by the user. Instead of doing a ttl probing, an Attack Request will be send directly to the server, and attack will launched base on the Attack

Request info. However, in this kind of attack, the attack main not always valid, due to the not valid ttl.

In getting the operator record, we only implement the AT&T post-paid plan data query in our demo, since it's wild use and support a high precision. And application-service model are employed in this attack, where the application periodically gets data from a query service running in background and posts it on UI. Decoupling these two functions makes it easier to develop, maintain, and reuse the modules.

## 5.4 PingPong Attack

### 5.4.1 Server Section

The Server part of Ping Pong attack is developed by using several python scripts. The intention of the server is to push the client to 3G from 4G repeatedly by making calls at a frequent intervals of time. However the server disconnects the call once it is established. In order to make calls , we have made use of skype calling services. Skype4Py , a python module/api is employed for the server to make repeated calls to the client. The calls will be repeated until it receives a UDP packet with a custom code in the payload.

### 5.4.2 Client Section

The main process to implement a PingPong attack is first need the user enter the phone's phone number and send it to the server by communicationSocket. While the server launch the skype calling attack, the ploting part will plot the cellular network status out. When the user wants to stop the attack, a stop message will send to server by pressing the stop button.

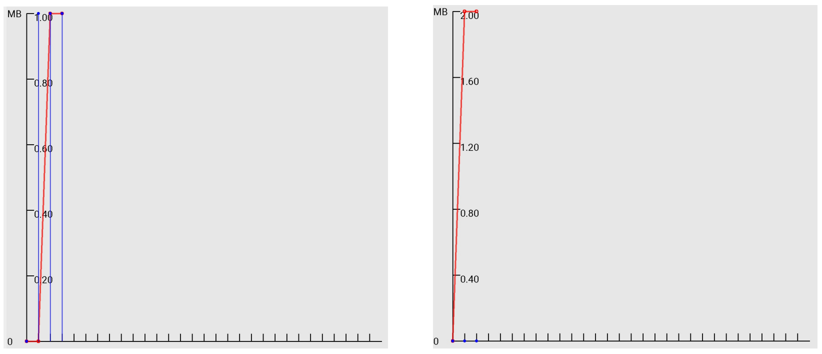
## 5.5 Back3G Attack

Back3G attack is the only attack in our demo can be launched without server's help. To launched the attack, we hard code the application to download a 1GB file from Internet, which is to keep a long enough time to track the download speed. While plotting the throughput of downloading process, a attack can be launched by pressing the attack button, which will go to dial a number out and make a call. In both CSFB and VoLTE scenario, the throughput would drop down sharply.

# 6 Results

In this section, the main results and verifications of demo are shown. After successfully deploying the server in an Linux server, these attacks can be easily implemented.

In TTL attack, AT&T family plane user are selected as the demo user and it has under the attack. We implement the attack in Columbus, Ohio, USA. The



(a) TTL=18 (b) TTL=16

Figure 2: Verification of TTL Attack, where 3MB TTL attack is launched

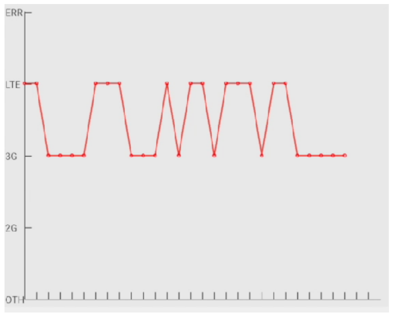


Figure 3: Verification of PingPong Attack

result shows that the valid TTL to launch the attack is 16 or 17. And also, a partial experiment is done in T-mobile, which the valid TTL is 17 or 18. In the Figure 2, we show out the verification of the attacks. A 3MB attack is launched in different TTL time. When TTL=18, which is shown in (a), local data usage, the blue line, is the same as the operator usage record, the red line, while when TTL=1, which is shown in (b), local data usage is equal to 0, and operator get 3MB volume in their system.

In PingPong Attack, CSFB scenario is implemented in our experiment. The server will launched a 10 times continuous call to the user, unless the user stop it manually, and the interval of the call is 10 seconds, which have the chance to sucks the user in 3G for a long time. However, it mainly to make the user change their network status frequently. Figure 3 shows the verification of the PingPong Attack.

In Back3G Attack, also CSFB is the target scenario. A 1.0 GB file is downloading to porvide enough time to track the throughput modification. In the experiment, after a call dial out, the throughput go down to zero, and will stay in a low throughput status, which sucks in 3G a while. Figure 4 shows the

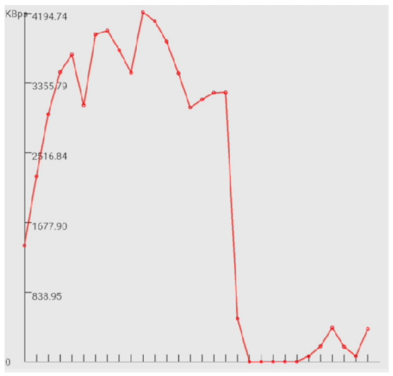


Figure 4: Verification of Back3G Attack

verification of the Back 3G Attack.

## 7 Conclusion

In order to propose the importance of insecure in MDC system, this project built an all-in-one demo (Android) to visualize and replay several mobile problem founded in these years. A framework to explore more extended attacks have been built in this demo too. Users would gain better and deeper understanding of cellular networks as they use this app anytime anywhere to collect data and compare the data difference. While the researchers can easily extend others attacks in our framework. This report analyzed root causes of three attacks, TTL attack, Ping-Pong attack, and Back 3G attack, which are the most visual problems in this area. It explained the implementation and evaluated the results of these attacks. Future work are needed because these loopholes still exist and cause customers overcharged. The secure of MDC system is a crucial part of cellular network system which required further improvements.

## Reference

[1] C.-Y. Li, G.-H. Tu, C. Peng, Z. Yuan, Y. Li, S. Lu, and X. Wang. Insecurity of voice solution volte in lte mobile networks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 316–327. ACM, 2015.

[2] C. Peng, C.-y. Li, G.-H. Tu, S. Lu, and L. Zhang. Mobile data charging: new attacks and countermeasures. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 195–204. ACM, 2012.

[3] C. Peng, C.-Y. Li, H. Wang, G.-H. Tu, and S. Lu. Real threats to your data bills: Security loopholes and defenses in mobile data charging. In *Proceedings*

*of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 727–738. ACM, 2014.

[4] C. Peng, G.-h. Tu, C.-y. Li, and S. Lu. Can we pay for what we get in 3g data access? In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 113–124. ACM, 2012.

[5] G.-H. Tu, C.-Y. Li, C. Peng, and S. Lu. How voice call technology poses security threats in 4g lte networks. In *IEEE Conference on Communications and Network Security (CNS)*, 2015.

[6] G.-H. Tu, Y. Li, C. Peng, C.-Y. Li, and S. Lu. Detecting problematic control-plane protocol interactions in mobile networks.

[7] G.-H. Tu, Y. Li, C. Peng, C.-Y. Li, H. Wang, and S. Lu. Control-plane protocol interactions in cellular networks. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 223–234. ACM, 2014.

[8] G.-H. Tu, C. Peng, H. Wang, C.-Y. Li, and S. Lu. How voice calls affect data in operational lte networks. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 87–98. ACM, 2013.