# Android Phone Application to Detect Malicious Cell Phone Spoofing

**Abstract:** Many phone users rely on caller ID to correctly identify the person who is actually placing a call to them. Before answering the call, it is difficult to determine if the caller ID has been faked, and thus whether or not the caller ID accurately identifies the caller. A caller ID spoof becomes malicious when the intent is to make the victim believe they are receiving a call from a trustworthy number when it fact they are not. The goal of the project is to develop an application for the Android phone that can correctly identify whether an incoming call is originating from the displayed caller ID number. This is accomplished through immediately calling back the displayed number and analyzing that call. Verification of the displayed number indicates to the callee that the number has not been spoofed. While several applications exist to spoof your caller ID, we were unable to find any that would detect the spoof. Thus our approach represents a unique way of identifying a caller ID spoof attempt.

## I. Introduction

Most of us rely on caller ID functionality to properly identify an incoming call. However, our telecommunications systems have made caller ID spoofing possible; there currently exists software for the iPhone, Android and Blackberry phones to accomplish this. (Castiglione, De Prisco, De Santis, p. 12) Caller ID Faker is a popular free application for Android. (*Caller ID Faker*) Android is a mobile platform put forth by the Open Handset Alliance. The platform itself is open source, giving developers and manufacturers flexibility in the design of new products. (*Open Handset Alliance)*

Some reasons for faking a caller ID are legitimate, such as a business with many phone lines using a Primary Rate Interface (PRI) to show only one main public number when an internal line is used to place a call. Other reasons are more deceptive, making a victim believe they are receiving a call from a trustworthy number when it may actually be a prankster, a private investigator or someone more sinister. A simple prank would be a parent pretending to be their child's friend after the parent and child have argued. Private investigators have also used PRI to hide their real phone number. (*Caller ID Spoofing*) A recent report from ABC news in Springfield, PA illustrates the potential danger of caller ID spoofing. On June 2, 2010, many Springfield and Delaware County police officers arrived to at a family home in response to a 9-1-1 call. However, this call was the result of 'swatting'- a telephone fraud in which "pranksters spoof the caller ID and appear to be calling" from inside or near the home in question. (Cuellar) This dangerous ploy often suggests that there is an armed person holding hostages, and thus often creates quite a large response by the police who are prepared to be facing a volatile

situation. The unaware victims are at risk of serious injury and there is the potential for other crimes to not receive the response they should because police are tied up responding to a prank.

No one wishes to believe that they will be the victims of a spoof, as a prank or as malicious attack, but one must recognize that humans are still the most attractive target to a criminal, even as technology grows more pervasive. (Griffin, Rackley p.1) In other words, it often is easier to illicit confidential financial information from a person than it is to break into the bank's system and find that information. Therefore, it is increasingly more important that an individual is not only aware of potential attacks, but also has the tools available to fend them off. The focus of this research was to develop such a tool for the Android telephone that will be able to detect whether or not an incoming call has had its caller ID spoofed.

In order to accomplish this, we investigated several options. The first option was an application that would immediately send an SMS message that contains a verification code to the displayed number. However, there are some drawbacks to this approach, as it would require the callee to answer the call and demand the verification code be announced. Alternatively, a client could be installed on both the callers and the callees phones in order for an automatic response to the generated via SMS; however, this could also potentially be spoofed by someone so inclined. Additionally, we investigated whether it would be possible to immediately call back the incoming number, and then based on the response code from the network, determine if we were currently experiencing an incoming call from that device. We initially examined SIP protocols but determined that they would not provide the kind of responses we would need in order to detect a spoofed call.

The paper is organized as follows: First, in section II, we discuss background information on caller ID, and present some other approaches to detecting if the caller ID has been spoofed. Then, in Section III, we present our selected approach and discuss details about the methodology we used. Section IV outlines the experimental set up for this research. We discuss specific experiments in Section V and their results in Section VI. Section VII details our conclusions from these experiments. We discuss our future work relating to this topic in Section VIII. Lastly, in section IX we acknowledge our supports and list the references for this paper.

## II. Background

### II.1 Caller ID Basics

Caller ID is a feature that transmits the calling party's number to the callee during call set up. It can be sent as either Single Data Message Format (SDMF) or Multiple Data Message

Format (MDMF). (*MicroEngineering Labs, Inc)* The main difference between the two being that in MDMF, the calling party's name is sent in addition to the telephone number from which they are calling. However, there is nothing to ensure that the number sent by a switch is actually where the call has originated. The displayable information in caller ID can come from "either a header field in SIP or a database from the caller's phone service provider"; the main issue with this being that in SIP, the displayable information is not authenticated. (Chow, et al p. 2-3) Thus the very information people often rely on to determine whether or not to answer the phone, has no authentication behind it, and should not be completely trusted.

It is clear that the burden of this authentication lies in the hands of the consumer. Phone companies are not set up to handle authentication of every subscriber. They are simply in the business of providing phone services. The financial burden to authenticate every subscriber would be overwhelming, particularly for VoIP services that may have subscribers from a diverse geography.

**II.2 Previous work**

While there is some evidence that this problem is being considered, at this point we found no working solutions. Cai discusses how a system to validate caller ID information would work. His system is designed to "process the originating node information and the call ID information to determine whether the call originated from the originating node". (Cai Patent application) This solution is one that would need to be implemented within the telephone network itself. It is designed to process information in a stage between the call being routed from the originating note to the terminating node. That is, processing would happen on the network after the caller has dialed but before the callee's phone receives the signal to ring. He also suggests that there may be several ways of comparing this information depending on the configuration of the network and what information is available to examine.

Another approach, put forth by Chow, Gustave and Vinokurov is the RealName Registry. They propose that telephone service providers collaborate with concerned businesses to establish separate registries for every jurisdiction potentially overlapping with trademark jurisdictions. The registry will be monitored by the members to ensure that each name in the list is unambiguous. Once registered, an entity would receive a certificate of authenticity. Then every time they place a call within the jurisdiction of that certificate the public key on the certificate would be compared to the private key during call set up, so that the certificate will display with their caller ID information. Alternatively, the authentication process could happen

on demand, at the initiation of one of the parties. The advantages here include not increasing call set up time, and not authenticating for unanswered calls or calls that go to voicemail. This authentication would also work in reverse, so that if someone needed to call their bank, they could be assured they had not misdialed or been given a false number.

While Chow, et al seem to have an approach that could work well for business and is "scalable and flexible by nature", we feel that there may be some hurdles to actually implementing such a registry. (p. 14) First, while spoofing is a serious issue, we are not convinced that businesses would be ready to invest the resources in registering themselves and then continuing to monitor the registry to which they belong. Secondly, Chow, et al admit that users are often the "weak link in the information security chain". (p. 14) It appears that this registry service is something the user would have to subscribe to through their local service providor. The problem with this approach is that many will not wish to pay for a service they feel they will not need. We all pay the 9-1-1 service fees on our phone bills because it is a service we hope to never need, but when it is needed, it could literally save lives. Having an authentication of our credit card company, bank or doctor may not seem as serious, particularly because people often feel nothing bad will every happen to them. There is also no discussion of validating individuals. While there are times that a single individual spoofing another individual is a childish prank between friends, this registry system does not seem to address the issue of 'swatting' discussed above. Finally, as with any system that involves software, there should be a discussion of how to prevent the system from being hacked. A criminal will always seek a way around a system, and if they can manage to hack into the system, perhaps to set for themselves the *keySertSign* bit of the *keyUsage* RealName certificate, then we will find ourselves right back in the situation of being easily spoofed. So while this RealName registry may hold promise, we feel that these issues need to be addressed.

The approach that we discuss here differs from both Cai's work and Chow, et al in that it does not require changes to existing telephony systems nor does it require users to certify themselves through a registry.

## III. Approach

In analyzing the possible approaches for our solution, we felt that it was important to keep the implementation strictly on the side of the callee. The SMS based solution and any other future solutions that require a client on both sides of the call are at risk for potential security breaches. These breaches could leave the callee at a greater risk because they may feel the dis-

played number has a greater chance of being accurate if it was verified through such a solution without stopping to consider the possibility that the software was altered on the other end of the call. Therefore we pursued a solution that would only need to be installed on the callee's telephone.

To accomplish this goal we aimed to develop an application that would detect any new incoming calls and immediately dial the incoming number. Since the Android platform has limited the telephony tools that are available through the SDK, we had to develop our application directly in the platform source code. We determined that modifications would be necessary to the CallNotifier.java class. This class is a module that listens for phone state changes and other events from the telephony layer. It is able to trigger resulting UI behavior (such as starting the ringer, the Incoming Call UI, writing call log entries, etc).

We designed our modifications to immediately answer an incoming call and place it on hold, bypassing the usual behavior of alerting the user to a new call through ringing and displaying the Incoming Call UI. At the time of initially answering the call, a Boolean variable, validateID is set to true. Then, upon the software detecting that there is a holding call, our modification is designed to place an outgoing call back to the number that was auto-answered. This step of placing an outgoing call automatically is also restricted to happen only if there is no other active call (foreground call). Once the automatic outgoing call is placed, validateID is set to false. At this point, *information about how the system responds to the signals for the outgoing call.*

Upon determining whether or not the original incoming call was spoofed, the outgoing call is disconnected. Then, as is typical behavior, the background call is automatically switched to be the active foreground call. The validateID variable was necessary because upon disconnecting the outgoing call, the CallNotifier class once again recognizes a change on phone state and would otherwise attempt to put the call on hold again. *At this point, ideally the phone would go through its usual behavior of ringing and displaying the Incoming Call UI with the new feature of also displaying whether or not the caller ID was verified. The reason for this behavior would be so that the user experiences no change in the way in which they answer the phone. For billing purposes, a callee will have been on the phone longer than just the time they will be actually talking with the caller, and they will also have an outgoing call charged against them. We feel that this is not too much of a burden to place on the callee, as unlimited service plans are available. (ie the price of security)*

## IV. Experimental Setup

We used the Eclipse Version 3.5.2 (*Eclipse.org Home*) along with Android Software Developers Kit Version 2.1 (*Android Developers*) to develop our initial approach. Included with the Android SDK is the Android Virtual Device (AVD) emulator and the Dalvik Debug Monitor Server (DDMS) that were used prior to deploying the source code on the phone test the system for correctness and stability. The Android source code is available online at http://source.android.com/source/index.html. All of our modifications were written using java 1.6. The phone itself is an Android xxxxxxx on the yyyyyy network.

For each experiment, we upheld the following conditions. Two Android xxx phones on the yyy network were used. Phone A was running the original source code out of the box while Phone B was running our modified source code. Additionally, both phones had Caller ID Faker installed. In each situation Phone A was located at XXX while Phone B was at YYY. In each case, there were no active or holding calls on either phone. Calls were placed at various times of the day. Some where intentionally spoofed while others were not as detailed below.

## V. Experiments

**Case 1:** Phone A calls Phone B without spoofing the number.
**Case 2:** Phone A calls Phone B, spoofing the number to be 555-4321.
**Case 3:** Phone B calls Phone A, without spoofing the number.
**Case 4:** Phone B calls Phone A, spoofing the number to be 555-6789.

## VI. Experimental Results

a. Outcomes of case 1
b. Outcomes of subsequent cases

## VII. Conclusions

*Case 1 illustrated that the modification of our source code was able to correctly detect the valid caller ID and display the information to the callee. Further, Case 2 allowed the modified source code to properly detect and display to the callee that the caller ID had in fact been spoofed. On the other hand, both cases 3 and 4 illustrate that Phone A running the original source code from the manufacturer was unable to determine whether or not the caller ID had been displayed, behaving as all other phones currently behave. We feel that these few cases illustrate that detecting a spoofed Caller ID is possible.*

*Discuss the costs of our solution (paying for connectivity longer than actually speaking time & paying for outgoing call) Compare those costs to other security costs many often seek such as home security monitoring systems and credit fraud monitoring systems.*

*We have shown that our approach works in a limited fashion. We are only able to detect the spoof when the spoofed call is received at a time that the phone has been IDLE; that is there*

*were no other active or holding calls at the time the call was received. And, as the solution was developed in the Android platform source, it would only be available once these modifications have been worked into the standard Android platform that is released to manufacturers. As a result, we strongly suggest that the Android community work towards integrating our solution into their platform.*

*While solving the issues of spoofing the emergency system was outside the scope of this project, the fact that we were able to develop a solution for the Android system brings us all one step closer to protecting each individual and may spur future developments for the phone system as a whole. Additionally, our approach has the possibility of being extended to other mobile phone platforms.*

## VIII. Future Work

The system as we have developed it still needs refinements. For one, currently when the outgoing call is placed to detect the spoof, the user interface displays the holding call and the active outgoing call. Ideally, the user interface would not indicate any sort of telephony action until the caller ID verification process had completed. At that time the phone should go through the normal process of indicating that there was a new incoming call, along with whether or not the caller ID was valid. Secondly, we would like to investigate the possibility of extending our application to also validate a call that is received as an incoming call waiting call. This presents some hurdles, as our application first requires answering the call, as well as the ability to place that call on hold and dial out. At this point, to our knowledge, we would need to investigate the possibility of having an active call (The original conversation), a holding call (the incoming call waiting call) and an outgoing call to validate the incoming call. We are not sure whether this is possible.

## IX. References/Acknowledgements

During the course of this work, I was supported by grants from the National Science Foundation CNS-0353637 and CNS-0649105. I would also like to thank the following people from the University of South Carolina: Dr. Wenyuan Xu and her graduate students, Hossen Mustafa and Miao Xu; REU program co-directors Dr. Caroline M. Eastman and Dr. John B. Bowles, and REU Program Coordinator Laura Boccanfuso.

## Bibliography

*Android Developers*. Web. 07 June 2010. <http://developer.android.com/index.html>

Cai, Yigang. Patent Application: Validating Caller ID Information to Protect Against Caller ID Spoofing. US. 3 July 2008.

*Caller ID Faker – Fake a Call !* Web. 07 July 2010 <http://calleridfaker.com/?gclid=CJrdrM-D2qICFZFV2godzkDVvg>

*Caller ID Spoofing - Everything You Ever Wanted To Know About Caller ID Spoofing - CallerIDSpoofing.info*. Web. 07 July 2010. <http://www.calleridspoofing.info/>

Castiglione, Aniello, Roberto De Prisco and Alfredo De Santis. "Do You Trust Your Phone?" Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2009. 50-61.

Chow, Stanley T., Christophe Gustave and Dmitri Vinokurov. "Authenticating Displayed Names in Telephony." Bell Labs Technical Journal 14.1 (2009): 267-282.

Cuellar, Dann. *Pranksters Terrorize Delco Family in 'swatting' Call*. WPVI-TV, Philadelphia, PA, 4 June 2010. Web. 14 June 2010. <http://abclocal.go.com/wpvi/story?section=news/local&id=7479233>

*Eclipse.org Home*. Web. 07 June 2010. <http://www.eclipse.org/>

"Get Involved | Android Open Source." *Welcome to Android | Android Open Source*. Web. 07 July 2010. <http://source.android.com/source/index.html>

"Industry Leaders Announce Open Platform for Mobile Devices." *Open Handset Alliance*. Web. 07 July 2010. <http://www.openhandsetalliance.com/press_110507.html>

"MicroEngineeringLabs, Inc. | Caller ID." *MicroEngineering Labs, Inc. - Tools and Resources for PIC® Microcontrollers*. Web. 8 July 2010. <http://melabs.com/resources/callerid.htm>