University of Crete

Computer Science Department

Robust Prevention of Dial Attacks

Alexandros Kapravelos

Master's Thesis

Heraklion, June 2010

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΠΟΔΟΤΙΚΗ ΑΝΤΙΜΕΤΩΠΙΣΗ ΕΠΙΘΕΣΕΩΝ ΤΥΠΟΥ DIAL

Εργασία που υποβλήθηκε απο τον

Αλέξανδρο Καπραβέλο

ως μερική εκπλήρωση των απαιτήσεων για την απόκτηση

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

Αλέξανδρος Καπραβέλος, Τμήμα Επιστήμης Υπολογιστών

Εισηγητική Επιτροπή:

Ευάγγελος Μαρκάτος, Καθηγητής, Επόπτης

Σωτήρης Ιωαννίδης, Ερευνητής Ινστιτούτου Πληροφορικής ΙΤΕ, Μέλος

Δημήτρης Νικολόπουλος, Αναπληρωτής Καθηγητής, Μέλος

Δεκτή:

Πάνος Τραχανιάς, Καθηγητής

Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Ιούνιος 2010

Robust Prevention of Dial Attacks

Alexandros Kapravelos

Master's Thesis

Computer Science Department, University of Crete

Abstract

The way we communicate nowadays has changed due to the advancement of Voice Over IP (VoIP) technology, which has enabled the interconnection of the Internet and the telephone network as Internet users can call landline or mobile devices through VoIP services. Although, this technology has many advantages and has been widely deployed, there are security concerns that yet have to be examined. The focus of this work is to explore the security properties that arise from making accessible telephone devices from the Internet through the use of VoIP.

We carry out attacks using Internet services that aim to keep telephone devices busy, hindering legitimate callers from gaining access. We use the term *DIAL (Digitally Initiated Abuse of teLephones)*, or, in the simple form, *Dial attack*, to refer to this behavior. We develop a simulation environment for modeling a Dial attack in order to quantify its full potential and measure the effect of attack parameters. Based on the simulation's results we perform the attack in the real-world. By using a Voice over IP (VoIP) provider as the attack medium, we manage to hold an existing landline device busy for 85% of the attack duration by issuing only 3 calls per second and, thus, render the device unusable. The attack has zero financial cost, requires negligible computational resources and cannot be traced back to the attacker. Furthermore, the nature of the attack is such that anyone can launch a Dial attack towards any telephone device.

Our investigation of existing countermeasures in VoIP providers shows that they follow an *all-or-nothing* approach, but most importantly, that their anomaly detection systems react slowly against our attacks, as we managed to issue tens of thousands of calls before getting spotted. To cope with this, we propose a flexible anomaly detection system for VoIP calls, which promotes fairness for callers. With our system in place it is hard for an adversary to keep the device busy for more than 5% of the duration of the attack. We also propose defenses on the client side, implemented as a fully functional call centre with the use of *Phone CAPTCHAs* to defend against DIAL attacks.

Supervisor: Evangelos Markatos
Professor

ΑΠΟΔΟΤΙΚΗ ΑΝΤΙΜΕΤΩΠΙΣΗ ΕΠΙΘΕΣΕΩΝ ΤΥΠΟΥ DIAL

Αλέξανδρος Καπραβέλος
Μεταπτυχιακή Εργασία
Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Περίληψη

Ο τρόπος με τον οποίο επικοινωνούμε σήμερα έχει αλλάξει λόγω της εξέλιξης της τεχνολογίας Voice over IP (VoIP), η οποία επέτρεψε τη διασύνδεση του Διαδικτύου και του τηλεφωνικού δικτύου, καθώς οι χρήστες του Διαδικτύου μπορούν να καλέσουν σταθερά ή κινητά τηλέφωνα μέσω υπηρεσιών VoIP. Παρά το γεγονός ότι η τεχνολογία αυτή έχει πολλά πλεονεκτήματα και έχει υιοθετηθεί ευρέως, υπάρχουν ανησυχίες για την ασφάλεια αυτής της διασύνδεσης, που πρέπει να εξεταστούν. Το επίκεντρο αυτής της εργασίας είναι να διερευνήσει την προβλήματα ασφάλειας που προκύπτουν καθιστώντας προσιτές τις συσκευές τηλεφώνου μέσω του Διαδικτύου με τη χρήση της τεχνολογίας VoIP.

Στα πλαίσια αυτής της εργασίας, έχουμε πραγματοποιήσει επιθέσεις μέσω Διαδικτύου, οι οποίες στοχεύουν στο να διατηρούν τις τηλεφωνικές συσκευές απασχολημένες, έτσι ώστε να αποτρέπονται οι θεμιτοί χρήστες από το να αποκτήσουν πρόσβαση. Χρησιμοποιούμε τον όρο *DIAL (Digitally Initiated Abuse of teLephones)*, ή, στην απλή μορφή, *επίθεση Dial*, για να αναφερθούμε σε αυτήν τη συμπεριφορά. Έχουμε αναπτύξει ένα περιβάλλον προσομοίωσης για την μοντελοποίηση μιας Dial επίθεσης, προκειμένου να ποσοτικοποιηθεί το πλήρες δυναμικό της και να μετρήσουμε την επίδραση των παραμέτρων της επίθεσης. Με βάση τα αποτελέσματα της προσομοίωσης πραγματοποιούμε την επίθεση ελεγχόμενα στον πραγματικό κόσμο. Με τη χρήση ενός Voice over IP (VoIP) παρόχου ως το μέσο της επίθεσης, καταφέρνουμε να απασχολήσουμε μια συσκευή απασχολημένη για το 85% της διάρκειας της επίθεσης με τη χρήση μόνο τριων κλήσεων ανά δευτερόλεπτο και έτσι καθιστάμε τη συσκευή άχρηστη. Η επίθεση έχει μηδενικό οικονομικό κόστος, χρησιμοποιεί αμελητέα ποσότητα υπολογιστικών πόρων και δεν μπορεί να εντοπιστεί ο επιτιθέμενος. Επιπλέον, η φύση της επίθεσης είναι τέτοια που ο καθένας μπορεί να ξεκινήσει μια Dial επίθεση προς οποιαδήποτε τηλεφωνική συσκευή.

Η δική μας έρευνα των υφιστάμενων αντισταθμιστικών μέτρων σε VoIP παρόχους δείχνει ότι ακολουθούν την προσέγγιση *όλα ή τίποτα*, αλλά το πιο σημαντικό είναι ότι τα συστήματα ανίχνευσης ανώμαλης συμπεριφοράς αντιδρούν αργά στις επιθέσεις μας, καθώς καταφέραμε να πραγματοποιήσουμε δεκάδες χιλιάδες κλήσεις, πριν γίνουμε αντιληπτοί. Για να το αντιμετωπίσουμε αυτό, προτείνουμε ένα ευέλικτο σύστημα ανίχνευσης ανώμαλης συμπεριφοράς για κλήσεις VoIP, το οποίο προωθεί δικαιοσύνη για τους θεμιτούς καλούντες. Με το σύστημά μας ως μέσο αντιμετώπισης, είναι δύσκολο ένας επιτιθέμενος να κρατήσει τη συσκευή απασχολημένη για περισσότερο από το 5% της διάρκειας της επίθεσης. Προτείνουμε επίσης τεχνικές άμυνας στην πλευρά του θύματος, τις οποίες τις υλοποιήσαμε ως ένα πλήρως λειτουργικό τηλεφωνικό κέντρο χρησιμοποιώντας τηλεφωνικά CAPTCHAs, το οποίο αντικρούει επιθέσεις τύπου DIAL.

Επόπτης Καθηγητής: Ευάγγελος Μαρκάτος
Καθηγητής

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω θερμά τον Ιάσονα Πολάκη και Ηλία Αθανασόπουλο για την ουσιαστική συμβολή τους στη περάτωση αυτής της εργασίας. Θα ήθελα επίσης να ευχαριστήσω όλα τα μέλη της ομάδας του DCS για την ευχάριστη συνεργασία μας αλλά και για το γεγονός ότι υπήρξαν όχι μόνο συνάδελφοι αλλά και φίλοι. Σε αυτό το σημείο θα ήθελα επίσης να ευχαριστήσω τον επόπτη καθηγητή μου κ.Ευάγγελο Μαρκάτο και τον Σωτήρη Ιωαννίδη για την άψογη συνεργασία μας και για την πολύτιμη καθοδήγηση τους κατά τη διάρκεια των ακαδημαϊκών μου χρόνων.

Θα ήθελα να ευχαριστήσω τους καλούς μου φίλους για τις υπέροχες αναμνήσεις που μου χάρισαν για τα χρόνια που πέρασα στο Ηράκλειο. Ακόμη περισσότερο, θέλω να ευχαριστήσω την Βίκυ Παπαβασιλείου που ήταν πάντα δίπλα μου να με στηρίζει σε αυτή τη προσπάθεια και που έδωσε άλλο νόημα στη ζωή μου από τότε που τη γνώρισα.

Τέλος, θέλω να ευχαριστήσω τους γονείς μου Νίκο και Νέλλη και την αδερφή μου Εύα που έκαναν τα αδύνατα δυνατά ώστε να μπορέσω να πραγματοποιήσω τα όνειρα μου.

Contents

List of Figures

List of Tables

1

Introduction

The Internet is a complicated distributed system that interconnects different kinds of devices and interfaces with other types of networks. Traditionally, computer security deals with attacks that are launched from Internet hosts and target other Internet hosts. However, the penetration of Internet services in everyday life enables threats originating from Internet hosts and targeting non Internet infrastructures. Such a non Internet infrastructure is the telephony network, a vital commodity.

The ever increasing number of households that adopt Voice over IP technology as their primary telephony system, demonstrates our shifting towards a digitally interconnected community. According to estimations, IP communication subscribers will reach more than 1.8 billion worldwide by

2013 [2]. While this new technology coexists with the old technology, new methods for their interaction emerge. Today, an Internet user can place calls to anywhere in the world reaching anyone that has a telephone device and take advantage of all characteristics inherent in such digital technologies, thus introducing new threats against traditional telephony systems.

In this thesis, we explore the feasibility of an attack using Internet services and targeting regular landline or cellular phones. We seek to characterize the parameter values that will make the attack effective and also the means to mitigate it. Our key contributions are the following:

Dial Attack. We develop an empirical simulation in order to explore the potential effectiveness of Dial attacks. Through the simulated environment we identify and quantify all of the attack's fundamental properties. Using experimental evaluation with existing telephone lines, we demonstrate that an *attacker manages to render an ordinary landline device unusable, holding it busy for 85% of the attack period by issuing only 3 calls per second* . The attack requires no financial resources, negligible computational resources and cannot be traced back to the attacker.

Defenses. We seek to reveal existing countermeasures through reverse engineering of real-world VoIP providers. Our findings suggest that current schemes are not efficient since they follow an *all-or-nothing* approach. We develop and analyze a server-side anomaly detection system for VoIP traffic, which significantly reduces the attack impact. With our defense system deployed, the attacker *can no longer hold the line busy for more than 5% of the attack period.* As telephone devices have no means of defense against Dial attacks, we incorporate phone CAPTCHAs in a call centre as a defense measure when countermeasures of VoIP providers fail to prevent attacks.

This thesis is organized as follows. We analyze our motivations in chapter 2. In chapter 3 we explain in detail how the SIP protocol works. In chapter 4 we present a threat model and a potential attack in a simulated environment. We carry out the attack against a real landline device in chapter 5. In chapter 6 we present existing countermeasures and introduce our anomaly detection system, together with experiments that show how effectively our system mitigates the attack. We also propose and develop a client-side solution to defend against DIAL attacks. Finally, we review prior work in chapter 7 and conclude in chapter 8.

2 Motivation

In this chapter we present the basic motivations that drove us to explore this area and led to the creation of this paper. We explore our motivation in terms of *goal* and *attack platform*.

Goal. The traditional communication through the telephone network has become an important commodity. Take into account, that over 300 billion domestic calls to landlines were served inside the US alone in 2005 according to the FCC [7]. Our argument is that *access to a telephone device is vital for humans*.

Considering the importance of the service, an adversary may target a telephone device in order to harm a user. Prohibiting users from accessing certain services has been done in the recent past. For example, a significant part of computer viruses disrupt Internet connectivity. The impact of such an attack can be enormous, either life threatening (targeting a fire-fighting station during a physical disaster), or financial (targeting a business, like pizza delivery, or hindering a bank to authenticate a transfer request [8]) or simple disturb someone.

Our research is composed by two complementary goals. The first goal is to find out if it is possible to render a telephone device unusable. We want to achieve this goal with no financial resources, negligible computational resources and without being traceable back to the attacker. The second goal is to design and build technologies for protecting users from attacks that target telephones. We want to achieve this goal with minimal deployment effort, minimal user interference and by using existing well-known technologies.

Attack platform. In order to achieve our goals we use VoIP providers as attack platforms against telephone devices. Our choice was driven by various reasons. First, we wanted an attack platform, which is affordable and easy to access. There are hundreds of free VoIP providers, which permit users to access any landline device with no cost at all or mobile devices with a minimal cost. Second, we wanted to be able to completely automate the attack and have enough flexibility in fine tuning the call placement. Most VoIP providers support the SIP protocol [24] which met our expectations. Third, we wanted to perform the attack anonymously. The very nature of VoIP technology allows a caller to hide his true identity. And finally, we wanted to launch the attack from a PC. The fact that our attack platform is already provided by the industry and anyone can use it to launch the attack motivates us highly to explore the area as a precaution from future exploitation of VoIP services.

One can argue, that parts of the attack described in this paper are well-known or can be carried out, manually, by performing an excessive amount of dialing. As far as the novelty is concerned, to the best of our knowledge, this thesis is the first one to perform and evaluate a real automated attack *directly* targeting a telephone device. As far as manual dialing is concerned, we already enlisted the four reasons, which drove us to select VoIP as the attack platform. These four reasons, reveal characteristics of a platform far superior to humans performing manual dialing.

3 Background

In this chapter we present an overview of VoIP technology and the SIP protocol to better understand the internals of a *Dial* attack.

Our work utilizes the SIP signaling protocol to set up the calls over the IP network. SIP is a text-based protocol similar to HTTP that follows a request-response structure, and can be used to set up or tear down sessions of any type. The content itself is typically transported over the Real Time Transport protocol (RTP) [25]. SIP messages are human-readable and are independent of the transport protocol used. For session initialization, the description of session characteristics such as the identifier of the initiating entity or the media content of the session, are encoded into a document using the SDP format [18]. The SIP protocol can be described as peer-to-peer since the SIP

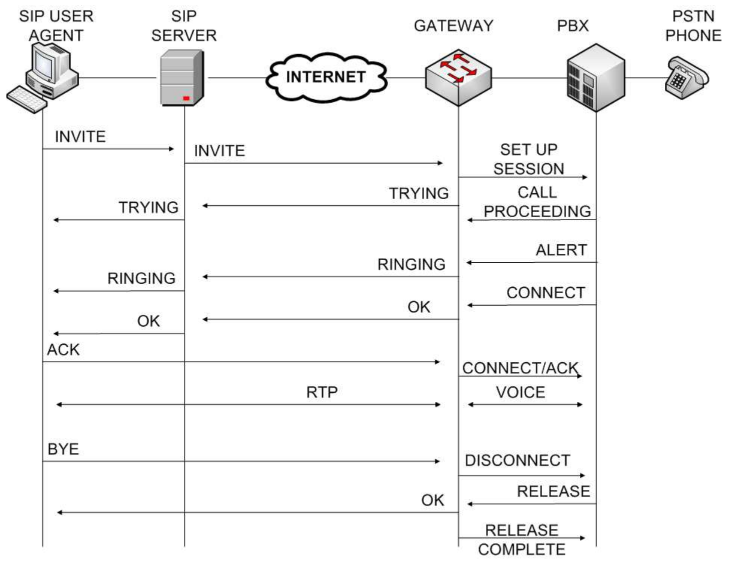

FIGURE 3.1: Steps in setting up and tearing down a SIP session (SIP status codes are emitted).

end users, called SIP User Agents, have two logical components: the User Agent Client that is responsible for sending request messages and receiving the responses, and the User Agent Server that receives requests and sends responses. Even though the simplest network configuration possible is two SIP user agents sending messages, typical networks also contain one or more of the following entities: a proxy server, a registrar or a redirect server. A proxy server is used for routing. Proxy servers forward session invitations to other proxy servers that are closer to the destination, until one is reached that knows the exact location of the destination user agent. They are also used to ensure policies. A registrar accepts REGISTER requests from SIP user agents and stores information regarding their current location (IP address, port, username) into a location database. A redirect server receives requests and replies with a list containing the current location for a specific user, as read from the location database created by a registrar. All SIP entities are identified through a SIP Uniform Resource Identifier (URI), which follow a form similar to that of email addresses: sip:username@domain.

We can see in detail how a session is set up and torn down between a SIP user and a PSTN landline. After registering with a SIP server, the SIP client is able to perform calls. To initiate a call, an `INVITE` message is sent to the SIP server that sets up the session with the PSTN landline. When the session is ready, the SIP client receives a ringing response from the SIP server. When the PSTN device answers the call, data is transported in real time. When the SIP client terminates the session, a `BYE` message is generated and sent to the other end.

4 Attack Overview

In this chapter we present the fundamental properties of the attack we developed. We start by describing our attack in detail; we specify the threat model, the adversary's overall goal and list all the assumptions we have made. We develop a simulated environment in order to carry out the attack virtually. Based on our findings in this chapter, we proceed and develop the actual attack prototype in the next chapter.

4.1 Attack Description

The goal of the attacker is to render a telephone device unusable with zero financial cost. This can be achieved by injecting a significant number of *missed calls* towards a victim telephone device. A call is considered missed, if it is hanged up prior to the other end answering it. By placing the calls correctly in the network, the attacker can keep the target continuously busy and, thus, prevent other users from accessing the telephone device. Even though many VoIP providers allow calls to landlines free of charge, we designed our attack in a way to be able to attack cell phones even if such calls are not free. By hanging-up the placed calls on time, the adversary manages to launch the attack cost free. Even if the target telephone device is answered the attack does not degrade, but rather augments, taking into account that the resource is still in busy state.

The attack in principle consists of a resource R, an attack medium M and calling modules. The resource represents a telephone device and has two states; it can either be available or busy. The attack medium simulates the behavior of a VoIP provider; it receives requests and queries the resource in order to acquire it. A calling module places such requests to the attack medium at a configurable rate.

The proposed attack is based on some important assumptions. First, we assume that M is unreliable, meaning that communication messages may be lost, dropped or delayed. However, we assume that all faults in M are stabilized in the long run. Thus, we do not implement message faults for M in the simulated environment. Second, we assume that R does not support direct querying, or at least it supports it partially. There is no way to directly retrieve all states of R. However, it is possible to implement detection by analyzing parts of the communication messages. Third, we assume that, when no attack is taking place, the calling rate follows a Poisson distribution ($\lambda = 10$). When an attack is taking place, the calling rate significantly varies from the Poisson distribution. Finally, we make no assumptions about the routing latency for call placement, i.e. the time it takes for a request to reach R through M, or the release time of R after call termination. Instead, we perform real experiments to collect representative approximations of these quantities (see next section).

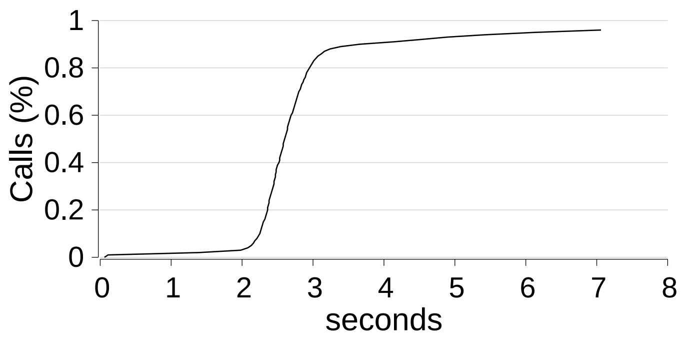

FIGURE 4.1: Cumulative distribution of routing latency times for call placement.

4.2 Simulation

Before starting experimenting with real calls in the wild, we performed a series of controlled experiments in a simulated environment. Based on the attack description we just presented, we developed a multi-threaded simulator where we instantiated a virtual calling module with an *aggressive behavior* to represent the attacker and one with a *non-aggressive behavior* to represent a legitimate caller. Then, we modeled the resource using a data structure that allows the module to obtain exclusive access with a certain probability. Each calling module behaves similar to a VoIP caller, *i.e.* it attempts to connect to the resource R through an attack medium M. In principle, M simulates a VoIP provider and R a telephone device.

In order to simulate the virtual calls in a realistic fashion we collected empirical values of durations from real call placement and hang-up operations. We issued 7,300 calls through a real VoIP provider over the time period of one week (see Figure 4.1). In this way, we collected representative routing latencies of call placements at various times and days of a typical week. The simulator maintains a pool with the 7,300 routing latencies and uses one, randomly, each time a virtual call attempt takes place. Unfortunately, we could not follow a similar approach for the hang-up operation, since it is hard to detect representative values for hang-up times of a real VoIP provider. However, we used the following approach, which we consider quite realistic. We injected pairs of call placements and hang-up operations in a real VoIP provider. We initially started injecting the pairs back-to-back. The result was that one of the two calls always reached the telephone device when it was in the busy state. In other words, the VoIP provider could not complete the hang-up operation of the first arrived call, before the second arrived. We started increasing the gap between the call pair, until we could measure that both calls had reached the telephone device in available state. We managed to successfully issue over 1,000 such call pairs with this property. The gap times ranged from 1 to 2 seconds. We consider this time window a realistic window for a hang-up operation. Thus, we modeled the virtual hang-up operation accordingly. Each virtual call hang-up operation takes from 1 to 2 seconds to restore R's state back to available.

Based on the above configuration we issued four 1-hour simulation runs, each one having an aggressive calling module placing virtual calls, with different intervals. We used intervals ranging from 0.01 to 5 seconds. Concurrently a legitimate module tried to acquire R following a Poisson distribution with $\lambda = 10$.

We examine the results of our experiments in terms of the aggressive calling module's success in acquiring resource R, the virtual call status distribution of the aggressive calling module and how many times the legitimate module succeeded in acquiring resource R. The rate of successful R acquisitions an aggressive thread managed to issue is presented in Figure 4.2. The best we could achieve was more than 39 acquisitions per minute. In the real-world, this result translates into more than 39 ringing calls per minute; a severe attack rate that would render the telephone device unusable. In Figure 4.3 we examine the call status distribution of

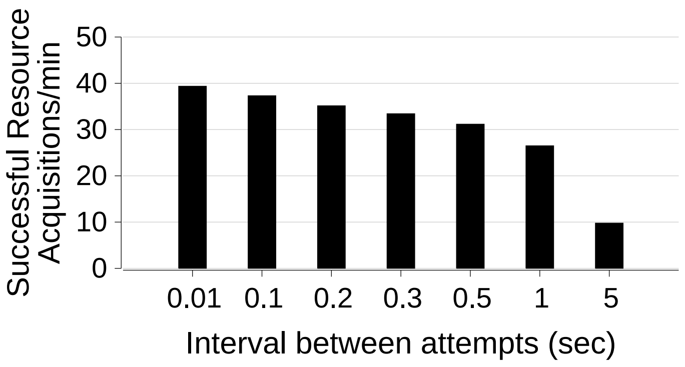

FIGURE 4.2: Rate of successful resource acquisitions managed by an aggressive calling module in simulation environment.

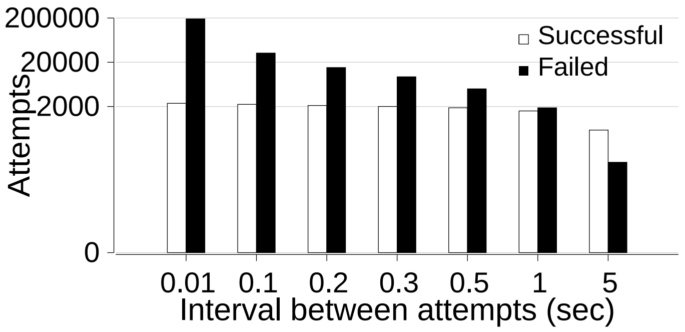

FIGURE 4.3: Distribution of all acquire attempts issued by an aggressive calling module in simulation environment.

all virtual call placements the aggressive calling module managed to issue. Observe that as the interval reduces, the amount of failures in acquiring R increases rapidly. Practically, there is no benefit in reducing the interval below 0.3 seconds.

The performance of the legitimate module is depicted in Figure 4.4. We also plot the results for the first 10 minutes of the experiment's duration in this case. First, observe that the legitimate module fails to acquire R for almost 85% of the simulation duration at the interval of 0.3 seconds,

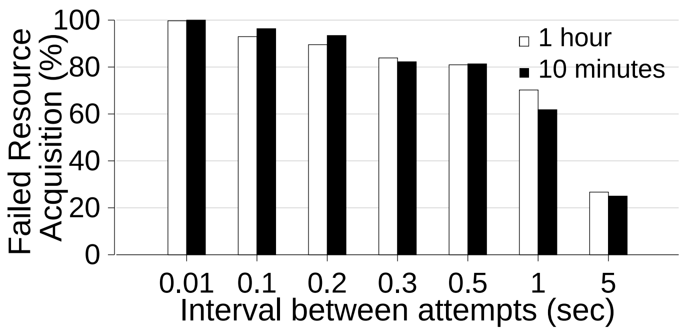

FIGURE 4.4: Percentage of failed resource acquisitions for the legitimate module which models a legitimate caller in simulation environment.

while the aggressive module issued less than 20,000 attempts. By reducing the intervals down to 0.01 seconds, the legitimate module is completely prevented from acquiring R, with the downside of requiring almost 200,000 more issued attempts to achieve just 15% more failed resource acquisitions compared to the interval of 0.3 seconds. Second, we can see that the first 10 minutes approximate the result of the full duration (1 hour) of the simulation, with tolerable error (from below 1% to 1.5%) in most cases. An exception occurs only in the case of the 1 second interval, where the difference is about 8%.

Our simulation experiments confirm our intuition for a potential threat against telephone devices. In addition to this, they highlight that only 3 calls per second are needed to render a device unusable and that the error between 1 hour and 10 minutes long experiments is tolerable.

5

Attack Evaluation

Based on the simulated studies we carried out in the previous chapter, we present an attack prototype. Our aim is to reach the performance we achieved in the simulated environment, using an existing system which tries to acquire an actual telephone device.

5.1 Attack Prototype

Our attack prototype implementation uses VoipDiscount [9] as an attack medium, which uses the Session Initiation Protocol (SIP) [24] for remote communication. As SIP is the most common used protocol among VoIP service providers, our prototype implementation is not limited to VoipDiscount but could be applied using any different provider.

We implemented caller modules, which communicate with M, in our case the VoIP provider, using the SIP protocol and exchange invite and termination messages. We used the Python programming language and the `pjsip`[1] library which provides an implementation of the SIP protocol. We developed two types of callers: (a) an attacker caller and (b) a legitimate caller. The attacker caller places calls one after the other, trying to keep the telephone device busy continuously. The legitimate caller places calls following the Poisson distribution ($\lambda = 10$).

Recall, from chapter 4, that we assumed that the resource does not support querying, or it supports partial querying. Indeed, the telephone device does not support querying and thus there is no easy way to track down the status of the device, i.e. if it is in ringing or busy state. Although, SIP supports querying the status of a placed call, many providers do not implement this feature. The one we used is among them. Specifically, we can retrieve that the line is busy, using a SIP operation, but we can not retrieve a ringing status. To overcome this issue we implemented a detector module, based on a Fast Fourier Transformation of the incoming audio signal. This way we analyze the frequency of the audio signal and detect a ringing tone when we observe signals at 420 Hz. Notice that this approach successfully recognizes the ringing tone, since the tone has a constant frequency. Having immediate access to the ringing status is vital for the attack, since we want to achieve the attack with zero financial resources. We want to keep the telephone device busy by injecting short time lived calls (i.e. missed calls). For the generation of a missed call, the call has to be terminated immediately after the first ringing tone.

This approach is more generic, as it is independent from the signaling protocol, in our case SIP, and always applicable, because having access to the audio signal when calling is essential. In Figure 5.1 we plot the spectrum of a ringing signal, as it is identified by the detector for a real call.

[1]PJSIP, `http://www.pjsip.org/`.

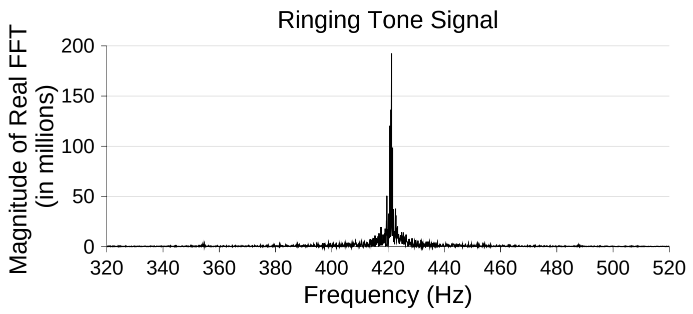

FIGURE 5.1: Ring signal frequency detection with Fast Fourier Tranformation.

5.2 Real World Experiments

We conducted several real world experiments over the period of eight months using a landline device located in our lab as a victim. For the presentation of this section we issued a series of runs over the duration of 1 week. This subset of runs is consistent with our overall experimental results. For each configuration we issued 6 runs each with a 10 minute duration with intervals ranging from 0.3 to 5 seconds. Recall from chapter 4 that the first ten minutes of each simulation run approximate the result of the full duration (1 hour) of the simulation, with tolerable error (from below 1% to 1.5%) in most cases. Thus we face the following trade-off: conducting more short-lived or less long-lived real world experiments. We chose the first approach, so as to be flexible enough to conduct a larger experimental base.

As was the case with the simulation, we are interested in three measurements: (i) the call rate of the attacker, (ii) the call status distribution of the attacker, and (iii) the probability for a legitimate user to acquire the resource, while an attack is taking place.

We present the call rate achieved by the attacker in Figure 5.2. Observe, that the results are highly consistent with the simulated ones (see Fig. 4.2). The adversary has managed to issue almost 40 ringing calls/minute for a

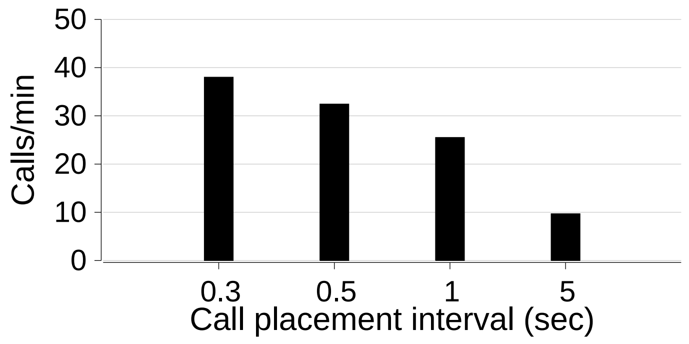

FIGURE 5.2: Rate of ringing calls managed by an adversary.

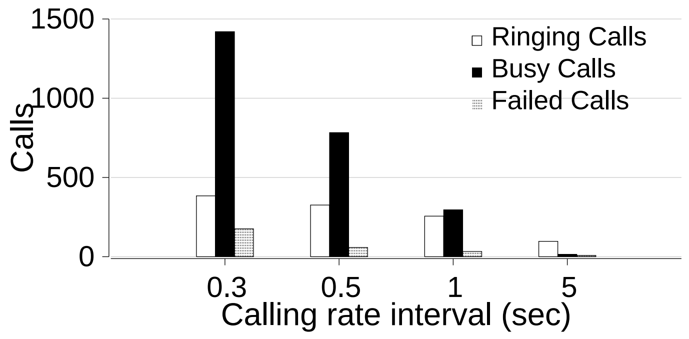

FIGURE 5.3: Distribution of all calls issued by an adversary.

calling placement interval of 0.3 seconds.

We present the distribution of all call attempts by the attacker in Figure 5.3. Again, the results are highly consistent with the simulated ones (see Fig. 4.3). Note, that as the call placement interval reduces, the fraction of busy calls increases, having a negative impact on the attack. Another side-effect of shorter call placement intervals is failed calls. A call is considered failed, when no ringing or busy status is identified after 10 seconds from call placement.

Finally, in Figure 5.4 we present the percentage of busy calls received by a legitimate caller, while the target telephone device was under attack.

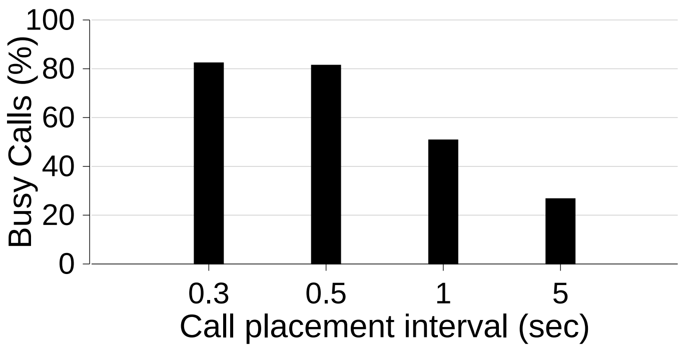

FIGURE 5.4: Percentage of busy calls received by a legitimate caller, while the target telephone device was under attack.

Observe, that the adversary managed to hold the target landline device busy for 85% of the attack duration, preventing access, for most of the time, to the legitimate caller.

5.3 Attack Impact

In our real world experiments, we managed to hold an existing landline busy for 85% of the attack duration, by issuing only 3 calls per second. By aggressively issuing calls, an attacker targeting the telephone centers of critical infrastructures such as police or fire-fighting stations and hospitals can completely disrupt their operation and create life threatening situations. By issuing dozens or hundreds of calls an attacker can hinder legitimate users from accessing the call centers of critical services. Taking into account that these services are vital to our society, *any* threat against them must be seriously considered, and mechanisms for protection should be designed and employed.

5.4 Attacker's Anonymity

An important aspect of *Dial attacks* is that they cannot be traced back to the attacker. This is achieved by having two layers of anonymity. The first is provided by the part of traditional telephony network, where only the VoIP provider's Caller ID is revealed. In our experiments, the Caller ID was *Unknown*, requiring the use of law enforcement in order to find the source of the calls. The second layer is the communication with the VoIP provider. The only way that the VoIP provider can track the attacker is by her IP address. In order to remain anonymous, the call requests must be placed from a safe IP address, *e.g.* through an anonymization proxy or with the use of a botnet.

6 Countermeasures

In this chapter we investigate existing countermeasures currently employed by VoIP providers. We present a study about Skype, a leading provider of VoIP services, VoipUser [10] and VoipDiscount [9], two representative VoIP providers. Based on our analysis, we propose and implement an anomaly detection system that promotes fairness to callers and is able to successfully mitigate the attack outlined in this paper.

Our key findings can be summarized as follows. First, existing countermeasures follow an *all-or-nothing* approach. A possible abuse results in permanently banning a user or even the target telephone device from the system. We describe, later in this section, that this policy is not only inefficient, but can also be part of further abuse, under certain circumstances.

Second, and most importantly, the existing countermeasures react slowly upon an abuse case. Indeed, we managed to issue tens of thousands of calls before getting spotted.

6.1 Existing Countermeasures

Skype. Skype is a popular VoIP provider with more than 400 million user accounts and capable of serving 300,000 simultaneous calls without any service degradation [1, 5]. Skype internally uses an anomaly detection system, whose technical details are not publicly available. In order to reverse engineer part of its logic, we used four different user accounts and three different landline devices. We performed experiments with very aggressive call initialization rates against our landlines. Eventually, all four accounts were blocked permanently and all three victim landline devices were permanently banned from the system. This means, that the victim landlines were further inaccessible by *any* Skype user. We refer to this policy as *all-or-nothing*, meaning that the anomaly system either permits full access or no access at all to the service. Skype maintains call history data for a period of six months as permitted by the applicable Luxembourg law [4]. We used the call logs from the Skype web site to create the call history of each account and telephone line.

In Figure 6.1 we present the cumulative time of the call history of each blocked Skype account. Our initial intuition was that Skype blocks our account when we pass a specific call-rate threshold. However, each Skype account got blocked when it exceeded a totally different threshold, indicating non-deterministic detection based on heuristics or human inspection of call logs. With the first account we placed more than *one hundred thousand* calls before the anomaly detection system spotted us. The other accounts were blocked by making a large number of calls in a very short time period. This is shown in Figure 6.1 by an almost vertical increase of at least fifteen

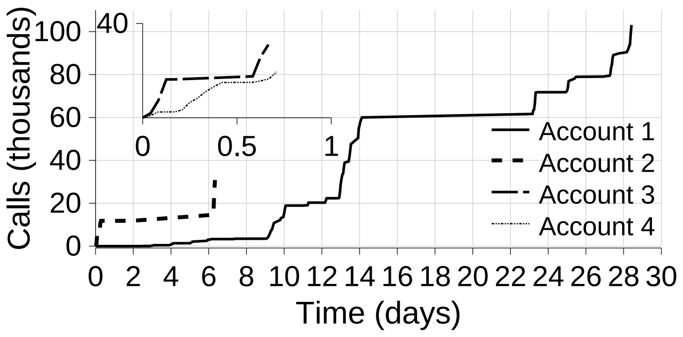

FIGURE 6.1: Call history of each Skype account, until it is blocked due to abusive behavior.

thousand calls.

In addition, Skype also blocked the landline telephone numbers which we used as victims. In Figure 6.2 we can see the call history of these numbers. The graphs terminate at the time the blocking actually happened. The Skype service permitted us to place more than 130,000 calls to the first line we used, before blocking it. The rest of the telephone lines we used were blocked as a result of more aggressive experiments.

We consider, that the *all-or-nothing* policy of Skype's anomaly detection is highly inefficient and, most importantly, enables further abuse. We proved that the slow reaction of the anomaly detection system allowed us to issue tens of thousands of calls. This would be catastrophic for any service that is based on telephone communication. We believe that the slow reaction is a fundamental result of the *all-or-nothing* approach. The penalty is so high (i.e. permanent block), that the anomaly detection system is triggered only during occasions where there is severe abuse. An adversary, could still carry out the attack in a more stealthy fashion. We also showed that an adversary can intentionally block certain devices from Skype. All she needs is to issue a vast amount of missed calls towards the victim device for it to be completely banned from the system.

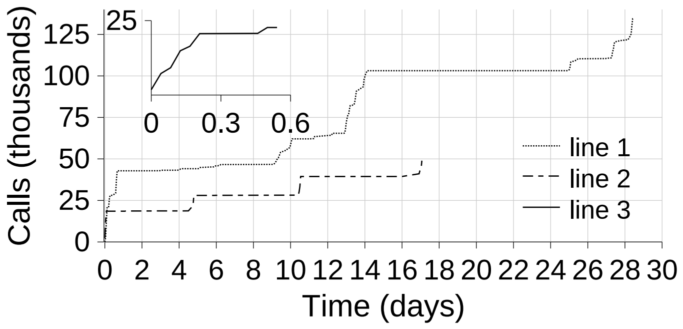

FIGURE 6.2: Call history of each telephone line targeted through Skype, until it is blocked.

Voipdiscount. During our experiments with the Voipdiscount provider we have not observed any countermeasures. We have used their infrastructure for multiple experiments, issuing hundreds of thousands calls, and Voipdiscount did not react to this behavior. We have been conducting experiments with their service for over 8 months without being warned or banned.

Voipuser. We speculate that Voipuser relies on manual inspection which is not effective and cannot provide adequate defense against such attacks. After a series of initial experiments we conducted, they blocked the accounts used, as well as all other accounts we had created; note that these accounts had not been used in the attack experiments. Account bans based on the correlation of the domain of the email addresses we used for the account registrations suggest a manual process of log inspection. However, our accounts were banned after the experiments had ended, proving the inability of manual countermeasures for the early detection of such attacks.

6.2 Server Side Countermeasures

Our system is based on a detection module and a policy enforcement module. We decided to implement the detection module entirely in software, using the well-known Intrusion Detection System (IDS), Snort [23]. As far as the policy enforcement is concerned, we have two options. We can either implement it in software or in hardware. For the first case, we can use the built-in firewall functionality of Linux operating systems, `iptables`. [1] However, this gives us poor flexibility in complex policies. On the contrary, the hardware solution gives as a range of functionalities employed by modern router devices. In order to easily perform an evaluation of various policies, we chose to use the Click router [20], which is a rich framework for testing router configurations. The Click router incorporates a wide range of elements for traffic shaping, dropping decisions and active queue management, which can also be found in most modern routers.

Detection Module. Snort is responsible for the detection. It handles user requests by monitoring all incoming traffic and flags flows that belong to hosts that initiate a large number of calls in a short amount of time. We further refer to this threshold as *abt (abuse behavior threshold)*, which is expressed in *invite*[2] requests per second. We have implemented, a Snort-rule similar to those for *port-scans* for detecting hosts that exceed *abt*. Whenever we have a Snort alert, the policy enforcement module is invoked, in order to mitigate the suspicious behavior.

Policy Enforcement Module. Policies are enforced over specific time windows. We refer to this quantity as *pew (policy enforcement window)*. Each policy applies an action to a host, that has been flagged suspicious by the detection module. We have implemented two different types of actions:

[1]For the core internals of Linux iptables, please refer to: `http://www.netfilter.org/`.

[2]An *invite* request in SIP is associated with a call placement.

mute and shape. The *mute action* drops all invitation messages and the *shape action* imposes a fixed rate of message delivery in a fashion that approximates a legitimate behavior.

It was tempting to also apply policies found in existing literature, as we reviewed in the related work chapter. In fact, we did try a policy for *selective dropping*, which drops packets according to the Random Early Detection (RED) algorithm [16].[3] As we mention in chapter 7, we explicitly refer to several existing countermeasures that alleviate a similar problem, such as using weighted queues before unsolicited traffic reaches a, potentially, victim interface or more strictly provisioning and partitioning resources [14, 16]. However, these mechanisms are designed with congestion avoidance in mind. As we have stated multiple times throughout this paper, our attack consumes negligible resources and does not lead to any congestion incidents. Nonetheless we artificially generated congestion in our detector in order to evaluate a *selective dropping* policy based on RED. However we consider that our assumptions, in regards to the conditions a real-world VoIP provider may experience, were very unrealistic and, thus, decided to provide no facts about this policy.

We implemented the *mute* action using `iptables` in Linux, by using Snort's plugin SnortSam [6]. We provide a hypothetical hardware implementation of both *mute* and *shape* actions using the emulation environment provided by Click. In Table 6.1 we summarize the policies we support, along with their notation.

6.2.1 Evaluation

In order to evaluate our anomaly detection system we performed our attack once again, but this time, both the attacker and legitimate caller were forced to pass their requests through our system. This was done at the network

[3] More precisely, we used the *gentle* version of RED [15], which is the RED implementation for Click.

Policy	INVITE behavior	Implementation
soft-mute	Drop	`iptables`
hard-mute	Drop	Click Router
hard-shape	Fixed rate	Click Router

TABLE 6.1: Policies supported by our anomaly detection system.

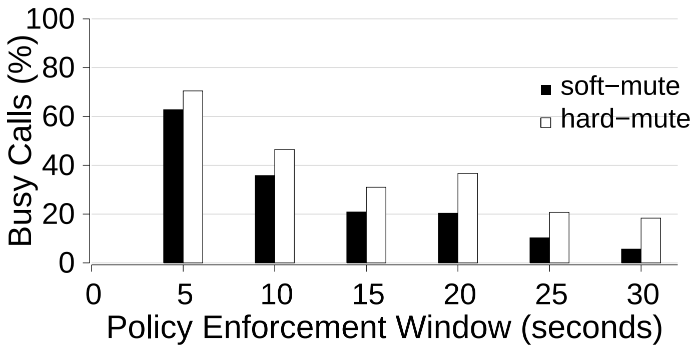

FIGURE 6.3: Attack mitigation for soft-mute and hard-mute policies for various *pew*.

level, by rerouting all communication messages through a gateway that acts as an anomaly detection system.

In order to eliminate false positives we decided to use a more tolerating *abt* value, equal to 10 invitation messages per 30 seconds ($abt = 10msg/30secs$). Notice, that although this decision leaves us with no false positives (indeed, we have measured zero false positives in all experiments), relaxing *abt* is negative for the mitigation result. The attacker can become more aggressive and still remain under *abt*.

In Figure 6.3 we depict the effects on the attack's firepower when our policies are enabled. Each policy is applied for a time duration equal to *pew*. Notice, we do not provide results for hard-shape values for any *pew*, since the hard-shape policy is enforced for the whole attack duration. This

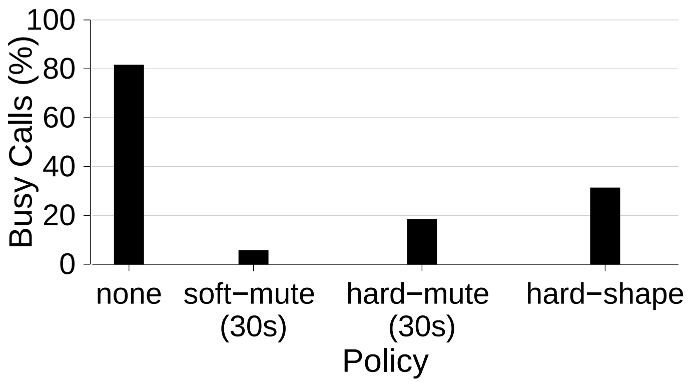

FIGURE 6.4: A comparison of all policies along with the original attack.

is not explicitly forced by our detector, but it stems from the fact that the attacker does not adapt to the policy, and the *pew* is always extended.

In Figure 6.4 we provide a comparison of all policies along with the original attack. For each policy we state the *pew* used inside parenthesis. Observe that the attack's firepower can be reduced to 5% using *soft-mute* or up to 30% using a more relaxed policy, *hard-shape*. We consider the *shape* policy more relaxed than the *mute*, since the suspicious host is not muted and thus the policy is more tolerable in enforcing restraints on false positives.

6.3 Client Side Countermeasures

In this chapter we explore client-side countermeasures: that is, countermeasures to be employed by the target telephone device. Telephone devices currently have no means of defense against the attack outlined in this thesis. Our goal is to enable potential targets to defend against this type of attacks regardless of the countermeasures that Voip providers may, or may not, incorporate. We are, to the best of our knowledge, the first to design, implement and test a fully functional system that can mitigate such attacks. Our solution is based on *Phone CAPTCHAs*. A CAPTCHA [29] is a challenge test that requires a response before an action can be performed and is used in computing to ensure that the request for the action is not automatically initiated by a computer. The goal is to prevent computer programs from performing certain actions that will lead to the degradation of quality of a certain service. Phone CAPTCHAs are specifically developed to be deployed with software implementations of private branch exchange systems. Even though CAPTCHAs are inherently inefficient against certain threat-models such as the laundry-attack [12], they are nonetheless utilized by several major vendors such as Google and Microsoft. Consequently, our solution, although not without weaknesses, can constitute a useful and efficient defense measure in many cases.

The rest of this chapter is organized as follows. First we list all components used for the deployment of our platform. We proceed and outline how our solution can cope with the discussed attack. Finally we enumerate various limitations of our solution and propose enhancements for our phone CAPTCHAs.

6.3.1 Architecture

The goal of our system is to protect landlines from the SIP-based attack we described in chapter 4. Furthermore, it can be used to block automated SPAM over IP Telephony (SPIT) calls [22] the number of which will continue to increase as VoIP technology becomes cheaper and widely adopted. Here, we focus on how to defend against the attack. Nonetheless, our system needs no modifications to filter-out automated SPIT phone calls. We will first describe the components that comprise our system and then how we model *Phone CAPTCHAs*.

Software. The core component of our platform is the Asterisk PBX, an open-source software implementation of a private branch exchange (PBX). Asterisk can deliver voice over a data network and inter-operate with the Public Switched Telephone Network (PSTN) so as to create an automated call center for any organization. It supports Interactive Voice Response (IVR) technology, can detect touch tones, that is dual-tone multi-frequency (DTMF) signaling, and respond with pre-recorded messages or dynamically created sound files. For Asterisk to work as a PBX, dial plans have to be created to control all the devices and users connected to the system. Configuration files are used to register devices and users, and to define actions to be performed for incoming and outgoing calls.

A native language is used to define contexts, extensions and actions. Devices and users are assigned to a context that defines their dial plan and, thus, restricts the extensions they may access and the calls they can commence. This can be used to enforce organization policies regarding access permission for user groups. A context can contain several extensions, and is structured as a sequence of lines, each belonging to a specific extension. Extensions consist of several ordered lines, where each line performs actions on known variables or executes one of the many applications available to Asterisk. Each line has the following components: an extension, a priority and a command with its parameters. An example dialplan context can be found in Appendix 8.

Hardware. For Asterisk to handle landlines, the host machine must be equipped with specialized hardware that connects it to the PSTN circuit. Depending on the hardware used, several landlines can be connected to the host and handled by Asterisk. For our implementation of the defense system we used the Digium TDM411B PCI card, a half-length PCI 2.2-compliant modular gateway card for connecting analog telephone stations and an analog PSTN line to a PC. For our implementation Asterisk was programmed to handle all incoming calls to a single landline.

Figure 6.5 shows a diagram of a call center incorporating Phone CAPTCHA technology to be deployed as a defense measure against the attack outlined in this paper.

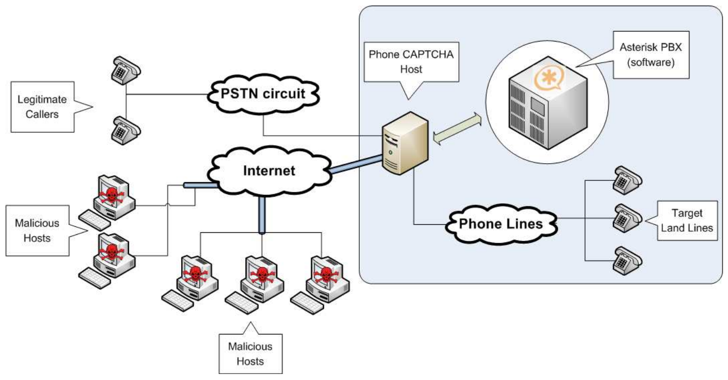

FIGURE 6.5: Diagram of a call center incorporating Phone CAPTCHA technology as a defense measure.

6.3.2 Phone CAPTCHAs

We use phone CAPTCHAs as a countermeasure to the Dial attack. Our phone CAPTCHA is a type of CAPTCHA crafted for use with the Asterisk PBX, but that can be deployed by any software PBX that supports IVR technology and call queues. When an incoming call is received, Asterisk places the call in a call queue. The caller then receives a phone CAPTCHA and has a limited time to respond to the CAPTCHA test using the phone's dial pad. This short time window is a way to make laundry-attacks against our system harder to carry out. The phone CAPTCHA test requires the caller to press a sequence of keys based on the instructions presented to him by a recorded message. For example, the phone CAPTCHA may ask the caller to `spell the word cat`. Or, it may ask the caller to `press the button which is the sum of 2 plus 3`.[4] If the caller provides the correct answer, Asterisk forwards the call to its destination as determined by the dial plan. Otherwise the call is dropped. This mechanism prohibits automated calls from binding to the end device and consuming resources,

[4]The specific questions may be customized to match the locale of each individual organization/user who deploys the CAPTCHAs

which could prevent legitimate callers from reaching the destination number. With our defense mechanism, attackers must incorporate automatic speech recognition software (ASR) in their effort to successfully launch an attack. Even though this is not a complete solution, it poses a significant obstacle for attackers when the CAPTCHAs are appropriatelly designed (see section 6.3.4). On the other hand, it is easy for legitimate callers to pass the phone CAPTCHA test.

If all telephone calls require solving a CAPTCHA before getting through, it may be annoying for callers. We recommend that CAPTCHAs be activated only when the telephone is suspected to be under attack. This way, callers will not always have to solve a CAPTCHA. Similar approaches have been adopted by Web services such as Google searches.

6.3.3 Limitations

Flooding the landline. Given that individual users and organizations have a limited number of landlines, an aggressive attacker may easily tie up all their incoming lines and overload their Phone CAPTCHA system. Fortunately, the Phone CAPTCHA system can easily be deployed by the victim's provider before it actually reaches the victim's own premises. In this way, the attacker can not reach the victim's premises and can not tie up the victim's landline without solving the CAPTCHA first. One might envision that such CAPTCHA tests may even be provided as a service by specialized companies which will have the capacity to handle thousands of incoming calls[5]. These companies may even have several centers distributed all over the globe, so as to easily tolerate attacks to any individual geographical region, much like content distribution networks operate today.

Breaking the phone CAPTCHA. Phone CAPTCHAs are vulnerable to

[5]This may be analogous to companies which today handle all incoming email of individual organizations so as to make sure that it is SPAM-free without overloading the organizations' own servers [3].

attacks that utilize ASR software to transcribe the audible information [26]. As aforementioned, traditional CAPTCHAs are subject to laundry attacks. While it would be far more complicated to deploy a laundry attack against phone CAPTCHAs, this limitation cannot be easily overcome. However, several parameters can be configured to make laundry attacks much more difficult. Such parameters include the available time-window, the number of attempts to provide the correct answer, the customization of the CAPTCHA's locale, etc.

Emergency situations. Even though the solution of a phone CAPTCHA is an easy task for a person under normal circumstances, it may become very difficult for people under extreme conditions. For example, people panicked due to fire, injury, or robbery, may not possess the mental lucidity to correctly answer the CAPTCHA. Thus, organizations, such as emergency services, which typically expect calls from people under such conditions, should employ other means of defenses against the described attacks. However, if such defenses are not available, e.g. due to budget restrictions, our Phone CAPTCHAs can still be part of a global solution, integrated with appropriate anomaly detection systems.

6.3.4 Phone CAPTCHA Enhancements

Simple phone CAPTCHAs may contain digits spoken by several speakers while other speakers that are audible in the background serve as noise that makes them more difficult for ASR software to break. However, as demonstrated by Tam *et al.* [26] this type of CAPTCHA can be broken. Therefore it was vital to enhance phone CAPTCHAs in such ways that their solution remains trivial for a person, but are robust against existing software that can pass phone CAPTCHAs. In this chapter we propose and implement enhancements that lead to the creation of more robust phone CAPTCHAs.

Expanding the vocabulary. Audio CAPTCHAs usually rely on a very limited vocabulary, namely digits, for security. By expanding the vocabulary to also contain complete numbers and mathematical operations, the training phase for ASR software becomes increasingly longer. A way to greatly extend the vocabulary, is to incorporate the use of words. Words have been widely used in the US to help people memorize telephone numbers by “translating" numbers into letters[6]. Based on that principle, phone CAPTCHAs can randomly select a word from a very large pool and ask the caller to spell it. This exponentially increases the complexity of the ASR's model and the duration of the training phase needed so as to be able to recognize such an immense vocabulary. It is important to note, that selected words should meet certain criteria, such as having a relatively small length, and being easy to spell, so as not to inhibit legitimate users from passing the test.

To further enhance the effectiveness of phone CAPTCHAS we intend to explore the incorporation of semantic information as a way of eliminating existing speech recognition software as a tool for breaking phone CAPTCHAs. For attackers to automatically solve CAPTCHA tests containing semantic information, their software must use machine learning techniques and knowledge representation in order to correctly answer the tests. Even questions that are trivial for humans to answer, such as “Which animal hunts mice?" or “What color is a red car?", demand sophisticated software. By issuing more elaborate questions, that are still answerable by most people, would require the attackers to have software far more advanced than what is widely available today. However, the system should select questions from a very large pool (ideally each question should be unique) so attackers can't identify previously issued questions and create a a dictionary that has the correct answer for each question.

[6]Who can forget for example the `1-800-FLOWERS` ?

6.3.5 User case study

Here we present the results of the user case study we conducted using our proof-of-concept countermeasure implementation. The goal was to measure the usability of our client-side solution and utilize user-feedback to improve phone CAPTCHA design. The 14 test subjects that participated in the study were students and staff from our campus, between the ages of 22 and 32. They were randomly separated into two user groups, the Informed Group and the Uninformed Group. The members of the first group were fully informed of the nature of the experiment, while those of the second group were simply asked to dial a phone number. The users of the first group knew that there would be a succession of 15 phone CAPTCHA tests, separated into 3 sets of tests. The first 5 tests would ask the user to spell a word, the next 5 to type the result of a simple mathematical calculation, while the next 5 would be a random succession of tests of the first two types.

User Group	Spelling Set	Calculation Set	Random Set
Informed Group	83	74	71
Uninformed Group	74	63	71

TABLE 6.2: Success rates(%) of the user study.

In Table 6.2 we see the results. As expected, the informed group achieved higher success rate (74-83%) than the uninformed one (63-74%) in the first two sets of tests, indicating that previous knowledge of the phone CAPTCHA type can lead to higher success rates.[7] In our experiment both user groups had the same success rate (71%) in the final test set. The

[7]Previous studies suggest that blind users were able to solve 43-46% of audio CAPTCHAS deployed by popular web services, while sighted users achieved 40-50% success rate on the same tests [13]. Using sophisticated interfaces, the success rate of the same set of blind users increased up to 70%.

users of both groups scored worse in the case of mathematical calculations. Most users stated that after the first couple of tests, it was easier to solve them. The phone CAPTCHA tests contained a significant amount of noise which led users to mistakes because they couldn't always make out the words. Moreover, since the Phone CAPTCHAs were in English and the test subjects had varying degrees of familiarity with the English language, this deployment represents an international deployment. We expect the success rates to be higher for a national deployment (i.e., where the language of the Phone CAPTCHAs matches the native language of the users). Nonetheless, the informed group successfully solved the spelling CAPTCHA tests 83% of the time, which leads us to believe that native speakers will be able to solve phone CAPTCHAs (that don't incorporate additional noise) with extremely high probability. This indicates that the robustness of phone CAPTCHAs must stem from the vastness of the vocabulary used and not the incorporation of additional noise. Furthermore, while the calculation phone CAPTCHA type offers only a marginal improvement in robustness, relatively to the basic type, it actually resulted in lower success rates. On the other hand, the spelling type CAPTCHAs had a much higher success rate and can utilize an immense vocabulary, making them far more robust against automated voice recognition attacks.

7 Related Work

In this work we use VoIP technology as an attack medium. Given its low access cost and its wide deployment, VoIP services have attracted a lot of attention [19]. For example, extensive research has been recently conducted on VoIP security. Wang *et al.* exploit the anonymity of VoIP calls by uniquely watermarking the encrypted VoIP flow [30]. Wright *et al.* investigate whether it is possible to determine the language of an encrypted VoIP conversation by observing the length of encrypted VoIP packets [31]. Zhang *et al.* in [32] exploit the reliability and trustworthiness of the billing of VoIP systems that use SIP [24]. In this paper we explore new ways for abusing VoIP services as well as identifying possible defenses to this abuse.

Research for attacks to the telephony network has been carried out in the past, mostly targeting cellular networks. For example, it has been shown that a rate of only 165 SMS messages per second is capable of clogging both text and voice traffic across GSM networks in all of Manhattan [27, 28]. Countermeasures to alleviate this problem are based on using weighted queues before traffic reaches the air interface, and/or more strict provisioning and partitioning resources after traffic leaves this bottleneck [14, 16]. Enck *et al.* demonstrate the ability to deny voice service by just using a cable modem [14]. They claim that with the use of a medium-sized zombie network one could target the entire United States. Their work also included suggestions on how to counter SMS-based attacks. Specifically, they call for the separation of voice and data channels, increased resource provisioning, and rate limits of the on-air interfaces.

As far as countermeasures are concerned, CISCO has filed a patent for a system that mitigates Denial of Service (DoS) attacks against call processing servers [21] in an IP network. Their system comprises of a network processing device and a controller that probabilistically decides whether a call will be accepted or rejected based on characteristics, such as the server's current load or whether the call originates from a previously authenticated source. This is highly efficient in scenarios where certain calls are of greater importance and must not be rejected even when the call processing servers are under heavy load. However, it is ineffective in cases where legitimate callers are not known in advance or can not be easily authenticated.

Last but not least, there are concerns in the research community about attacks that threaten the operation of emergency services. Aschenbruck *et al.* report that it is possible to peer VoIP calls to public service answering points (PSAP) [11]. This peering can have grave implications because it makes it possible to carry out DoS attacks against emergency call centers. In their work they monitored calls from a real PSAP of a fire department which serves about one million people. During emergencies the PSAP received approximately 1100 calls per 15 minutes. These calls overloaded the PSAP and the authors suggested that the high call-rate was the result of citizens constantly redialing until they got service. In their follow-up publication Fuchs *et al.* show that under heavy load at the same PSAP, up to half of the incoming calls were dropped [17].

8 Conclusion

In this paper we perform an extensive exploration of Dial attacks. Initiated by our theoretical findings, we implement a prototype and carry out the attack in the wild proving that an adversary can keep a telephone device in busy state for 85% of the attack duration by issuing only 3 calls per second. Our attack requires zero financial resources, negligible computational resources and cannot be traced back to the attacker. Considering the severity of such a threat, we explore already employed countermeasures and conclude that current VoIP infrastructures employ countermeasures based on an *all-or-nothing* approach, react slowly to possible abuse or offer no protection at all. As these defenses appear inefficient we propose an anomaly detection system for VoIP calls and demonstrate that it can mitigate Dial attacks and prevent an adversary from holding a telephone device busy for more than 5%. Furthermore, we designed and built a countermeasure architecture that effectively protects a landline against VoIP-based DoS attacks. We introduced the notion of *Phone CAPTCHAs*, a type of audio CAPTCHA designed to be utilized by a software PBX, that prevent automated calls from reaching the end devices.

Bibliography

[1] Ebay Inc. FQ 2008 results. `http://investor.ebay.com/results.cfm`.

[2] IDC Predicts more than 1.8 billion Worldwide Personal IP Communications Subscribers by 2013.
`http://www.idc.com/getdoc.jsp?containerId=219742`.

[3] MessageLabs. `http://www.messagelabs.com/`.

[4] Skype End User License Agreement. `http://www.skype.com/legal/eula/`.

[5] Skype Fast Facts, Q4 2008.
`http://ebayinkblog.com/wp-content/uploads/2009/01/skype-fast-facts-q4-08.pdf`.

[6] Snortsam. `http://www.snortsam.net`.

[7] Statistics of Communications Common Carriers 2005/2006 Edition.
`http://hraunfoss.fcc.gov/edocs_public/attachmatch/DOC-282813A1.pdf`.

[8] Thieves Flood Victim's Phone With Calls to Loot Bank Accounts. `http://www.wired.com/threatlevel/2010/05/telephony-dos/`.

[9] Voipdiscount. `http://www.voipdiscount.com`.

[10] Voipuser.org. `http://www.voipuser.org`.

[11] N. Aschenbruck, M. Frank, P. Martini, J. Tolle, R. Legat, and H. Richmann. Present and Future Challenges Concerning DoS-attacks against PSAPs in VoIP Networks. *Proceedings of International Workshop on Information Assurance*, 2006.

[12] E. Athanasopoulos and S. Antonatos. Enhanced captchas: Using animation to tell humans and computers apart. In *Communications and Multimedia Security*, 2006.

[13] J. P. Bigham and A. C. Cavender. Evaluating existing audio captchas and an interface optimized for non-visual use. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, New York, NY, USA, 2009. ACM.

[14] W. Enck, P. Traynor, P. McDaniel, and T. L. Porta. Exploiting Open Functionality in SMS Capable Cellular Networks. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05), Alexandria, Virginia, USA*, 2005.

[15] S. Floyd. Recommendation on Using the "gentle" Variant of RED. 2000. `http://www.aciri.org/floyd/red/gentle.html`.

[16] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1993.

[17] C. Fuchs, N. Aschenbruck, F. Leder, and P. Martini. Detecting VoIP based DoS attacks at the public safety answering point. *ASIACCS*, 2008.

[18] M. Handley, V. Jacobson, and C. Perkins. SDP: Session Description Protocol. RFC 4566 (Proposed Standard), 2006.

[19] A. D. Keromytis. A Look at VoIP Vulnerabilities. *USENIX ;login: Magazine*, 35(1), February 2010.

[20] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 2000.

[21] D. R. Oran. Method and apparatus for performing denial of service for call requests, 2003. US Patent 7380010.

[22] S. Phithakkitnukoon, R. Dantu, and E.-A. Baatarjav. VoIP Security: Attacks and Solutions. *Inf. Sec. J.: A Global Perspective*, 17(3), 2008.

[23] M. Roesch. Snort - lightweight intrusion detection for networks. In *LISA '99: Proceedings of the 13th USENIX conference on System administration*. USENIX Association, 1999.

[24] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), 2002. Updated by RFCs 3265, 3853, 4320, 4916.

[25] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), 2003.

[26] J. Tam, J. Simsa, D. Huggins-Daines, L. von Ahn, and M. Blum. Improving Audio CAPTCHAs. In *Symposium On Usable Privacy and Security (SOUPS) 2008*, 2008.

[27] P. Traynor, W. Enck, P. McDaniel, and T. L. Porta. Mitigating Attacks on Open Functionality in SMS-Capable Cellular Networks. *12th annual international conference on Mobile computing and networking*, 2006.

[28] P. Traynor, P. Mcdaniel, and T. L. Porta. On attack causality in internet-connected cellular networks. In *USENIX Security Symposium*, 2007.

[29] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. *CAPTCHA: Using Hard AI Problems for Security*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2003.

[30] X. Wang, S. Chen, and S. Jajodia. Tracking anonymous peer-to-peer VoIP calls on the internet. In *CCS '05: Proceedings of the 12th ACM conference on Computer and Communications Security*, 2005.

[31] C. V. Wright, L. Ballard, F. Monrose, and G. M. Masson. Language identification of encrypted VoIP traffic: Alejandra y Roberto or Alice and Bob? In *SS'07: Proceedings of the 16th USENIX Security Symposium*, Berkeley, CA, USA, 2007. USENIX Association.

[32] R. Zhang, X. Wang, X. Yang, and X. Jiang. Billing attacks on SIP-based VoIP systems. In *WOOT '07: Proceedings of the first USENIX Workshop On Offensive Technologies*.

Appendix

Example Asterisk dialplan

```
[incoming-calls]
exten=>s,1,Ringing
exten=>s,2,Wait,2
exten=>s,3,Playback(submenuoptions)
exten=>s,4,Read(INPUT||3)
exten=>s,5,Goto(incoming,\$INPUT,1)
exten=>11,1,Dial(SIP/Jason)
exten=>22,1,Dial(SIP/Alex)
exten=>i,1,Playback(invalid)
exten=>i,2,Hangup()
```

An example context for incoming calls to a system with two registered SIP users. Extension "s" is the starting extension for this context. First, the dial plan instructs asterisk to let the phone ring for two seconds before answering. Then an audio file is played informing callers of the extension that they must press to reach a certain user. The system reads the user's input, through DTMF signaling, and executes the first action of the matching extension, if one exists. Subsequently, the call is forwarded to the corresponding user. If no valid extension exists, the invalid extension "i" is invoked and an error message is played before terminating the call. For asterisk to handle landlines, the host machine must be equipped with specialized hardware that connects it to the PSTN circuit. Depending on the hardware used, several landlines can be connected to the host and handled

by asterisk.