# Digital is Calling the Analog: Robust Prevention of Dial Attacks

Alexandros Kapravelos, Iasonas Polakis,
Elias Athanasopoulos, Sotiris Ioannidis, and Evangelos P. Markatos
Institute of Computer Science,
Foundation for Research and Technology Hellas, Greece
email: {kapravel, polakis, elathan, sotiris, markatos}@ics.forth.gr

TR-399, October 2009
FORTH-ICS

**Abstract**

We carry out attacks using Internet platforms that aim on keeping telephone devices busy, denying users any access. We refer to this behavior using the term *DIAL (Digital Initiated Abuse of teLephones)*, or, in the simple form, *Dial attack*. We develop an intuitive simulation environment for modeling a Dial attack in order to identify its critical characteristics. Based on the simulation's results we perform the attack in the real world. By using a VoIP provider as the attack media, we manage to hold an existing landline device busy for 85% of the attack's duration and thus render the device practically unusable. The attack has zero cost and requires negligible computational resources. Furthermore, as we show, anyone can practically launch a Dial attack towards any telephone device.

However, in this paper, our primary goal is to protect telephones from Dial attacks. First, we investigate existing countermeasures in VoIP providers and show that they follow an *all-or-nothing* approach, but most importantly, their anomaly detection systems react slowly against our attacks. We managed to issue tens of thousands of calls before getting spotted. Second, using existing software technologies, Snort and Click, we present a flexible anomaly detection system, which promotes fairness to the callers. With our system in place it is hard for an adversary to keep the device busy for more than 5% of the attack's duration.

## 1 Introduction

The Internet is a complicated distributed system that interconnects many 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 Internet hosts. However, the penetration of Internet services in everyday life enables potential threats originating from the Internet 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 almost half a billion worldwide by 2012[1]. 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; one can also take advantage of all the characteristics inherent in such digital technologies and introduce new threats against traditional telephony systems.

In this paper, we explore the feasibility of a *Dial (Digital Initiated Abuse of teLephones) attack*. An attack originating from the Internet, carried out using a VoIP provider and targeting regular landline or

cellular phones. The attack aims on keeping a victim telephone device busy by injecting a large fraction of missed calls towards it. We seek to characterize the properties that will make the attack effective, but most importantly the means to mitigate it.

**Methodology.** As far as the first part is concerned, our methodology is the following:

*Analytic Model.* We derive an analytic model of a Dial attack based on its fundamental properties. We assume that an adversary is using an attack media that supports invitation and termination messages, just like an ordinary VoIP provider supports. Based on this, we formulate the basic goal of an attacker, which is practically to inject a vast amount of missed calls against a telephone device through a hypothetical VoIP provider.

*Simulation.* We explore the impact of a Dial attack in a simulated environment. We represent a telephone device using a programming `lock`. We use two families of threads which try to acquire the lock: (a) aggressive threads which follow attempt rates similar to the rate of an attacker injecting missed calls towards a telephone device and (b) threads following a Poisson distribution resembling a legitimate human trying to place a call. All lock-acquire and lock-release operations are modeled based on experimental measurements with real-world VoIP providers.

*Real-World Experiments.* We carry out Dial attacks using real-world VoIP providers. All experiments are tuned up according to the results of our simulated runs. In all real-world experiments an attacker competes with a legitimate caller to reach an existing landline. We measure the availability of the telephone device as experienced by the legitimate caller. Lower availability implies a higher attack impact.

As far as mitigation is concerned, our methodology is the following:

*Reverse Engineering.* We perform a series of reverse engineering experiments in order to reveal existing countermeasures employed by real-world VoIP providers. We aggressively place hundreds of thousands of calls in order to determine how VoIP anomaly detection systems work. We show that current schemes follow an *all-or-nothing* approach, but most importantly, they react slowly against our attacks. We managed to issue tens of thousands of calls before getting spotted.

*Dynamic Mitigation.* We design, develop and evaluate a dynamic system for mitigation of Dial attacks based on existing software technologies, namely Snort[14] and Click[10]. We test a series of different configurations in order to determine the effectiveness of our system during a Dial attack. We perform all real-world attacks again, but with our system in place.

**Contributions.** Our key contributions span along the *Dial Attack* and its *Defenses*:

*Dial Attack.* We develop an analytical model in order to explore the Dial attack space. Through a simulated environment we identify and quantify all of the attack's fundamental properties. Using experimental evaluation with real telephone lines, we demonstrate that an *attacker manages to render an ordinary landline device practically unusable, by holding it busy for 85% of the attack period.* The attack requires no financial resources and negligible computational resources.

*Defenses.* We seek to reveal existing countermeasures through the reverse engineering of real-world VoIP providers. Our findings suggest that current schemes are not efficient since they follow an *all-or-nothing* approach. By using well known software technologies (such as Snort[14] and Click[10]) we develop and analyze a server-side anomaly detection system, which significantly reduces the attack impact. The attacker *can no more hold the line busy for more than 5% of the attack period.*

**Organization.** This paper is organized as follows. We discuss in detail our motives in Section 2. In Section 3 we present a threat model and a potential attack in a simulated environment. We carry out the attack against a real landline device in Section 4. The rest of the paper is devoted to countermeasures. In Section 5 we present the results of a short reverse engineering effort, which aims to reveal the anomaly detection system employed by popular VoIP providers. We present our anomaly detection system, which builds on Snort and Click, in Section 6. Finally, we highlight our future steps for a client-side defense system in Section 7. We review prior work in Section 8 and we conclude in Section 9.

## 2 Motivation

In this section we present the basic motives that drove us to explore this area and led to the creation of this paper. We explore our motivation in terms of *goal* and *media*.

**Goal.** We view telephone devices as the *user desktops* of our age. Smart phones and modern mobile phones allow a user to connect to her contacts using the Internet or a telephone network. We argue that these devices can be seen as laptop replacements for basic needs such as exchanging e-mails and surfing the Web, but also as a gateway to traditional communication practices: using the telephone network to reach your "buddies". This traditional communication practice is 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[2].

Our thesis 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.

Our research is composed of 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 zero financial resources and negligible computational effort. We also want to keep the attacker anonymous. 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.

**Media.** 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 enable 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[15] 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. This is mainly driven by the fact that an adversary may own a BotNet, which can be used to launch the attack.

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 paper 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 enumerated the four reasons, that 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 Attack Overview

In this section we present the fundamental properties of the attack we developed. We start by defining our attack model; we list all the assumptions we have made, we specify the threat model and the adversary's

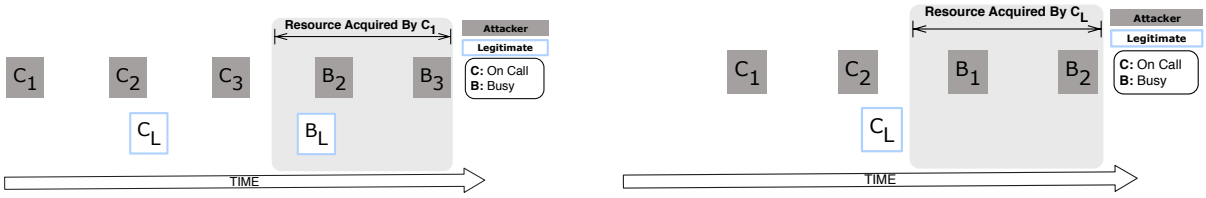


Figure 1: Schematic representation of a Dial attack. The attacker's attempts (gray boxes) are competing with legitimate attempts (white boxes) in order to acquire the resource. The term *attempt* in our context can be either a thread trying to acquire a lock or a real network VoIP packet trying to place a call in a telephone device. In the left figure the attacker's attempt has acquired the resource, but all further attempts inside this timeframe (boxes with 'B') will fail, as the resource will be busy. In the right figure the legitimate attempt has managed to be placed in the correct inter-gap of two attacker's attempts.

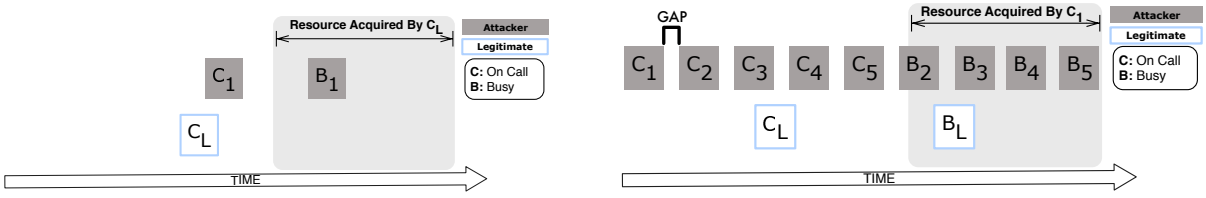


Figure 2: Schematic representation of a Dial attack. The attacker's attempts (gray boxes) are competing with legitimate attempts (white boxes) in order to acquire the resource. The term *attempt* in our context can be either a thread trying to acquire a lock or a real network VoIP packet trying to place a call in a telephone device. In the left figure a legitimate attempt has managed to acquire the resource, forcing the attacker's attempts inside this timeframe (boxes with 'B') to fail, as the resource will be busy. In order to reduce the probability of this case, the attacker has to place attempts which have minimal inter-gap with each other (see right figure). Our simulation experiments suggest that an efficient inter-gap period is 0.3 secs. Reducing the inter-gap period, also increases the failed attempts for the attacker.

overall goal. We develop a simulated environment in order to carry out the attack virtually. Based on our findings in this section, we proceed and develop the actual attack prototype in the next section.

### 3.1 Attack Model

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, when it is hanged up prior to being answered. 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 [11], 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 succeeds in launching the attack cost free. [1]

[1]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. One can argue that the attack does not remain cost free, but note that the fraction of answered calls will always be negligible compared to the unanswered ones.

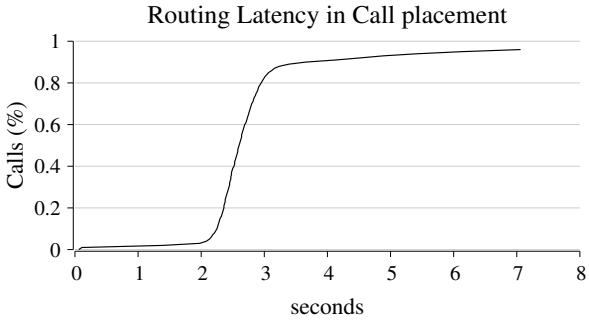


Figure 3: Representative routing latency times as collected from a series of experiments involving a real VoIP provider. We depict over 7,300 samples collected over the period of one week using a cumulative distribution function. Note, that a typical call requires about 2 to 3 seconds to be served.

We introduce the following terms in our model. We denote the attack media as $M$, which is responsible for transmitting communication messages, noted as $C$. Our model has invitation messages, $C_i$, and termination messages, $C_t$. We also denote with $R$ the resource. The resource has two states; it can either be available or busy.

The attack aims in keeping $R$ in the busy state. In order to do so, the attacker transmits a $C_i$ through $M$. Upon taking the resource, the attacker transmits a $C_t$ through $M$ in order to release it. Thus, $R$ is in the available state for short time windows. We denote $C_i(n)$ and $C_t(n)$ the arrival distributions in time domain of $C_i$ and $C_t$, respectively, where $n$ denotes the sequence of sessions. For example, session $n = 0$ is initiated by $C_i(n = 0)$ and terminates with $C_t(n = 0)$. With this notation, the resource $R$ is in busy state for the time period of: $\Delta t(n) = C_t(n) - C_i(n)$, for a given $n$. During an attack, the resource $R$ is in busy state for:

$$T_{busy} = \sum_{n} \Delta t(n), n \in N. \tag{1}$$

Optimally, we want to maximize (1) by sending a large number of coupled $C_i$ and $C_t$ messages.

We enlist some important assumptions. First, we assume that $M$ is unreliable. That means 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, while there is no attack taking place, $C_i(n)$ follows a Poisson distribution ($\lambda = 10$). While there is attack taking place, $C_i(n)$ significantly varies from the Poisson distribution. Finally, we make no assumptions about the routing latency for $C_i$, i.e. the time it takes for a $C_i$ to reach $R$ through $M$, or the release time for $C_t$, i.e. the time it takes for a $C_t$ to reach $R$ and release it through $M$. Instead, we perform real experiments to collect representative approximations of these quantities (see next section).

### 3.2 Simulation

Based on the attack model we just presented, we developed a simulated environment. The resource, $R$, is simulated using a programming `lock`. An attacker and a legitimate user are simulated using threads that try to acquire the lock. We assume that there is a detection module and different threads can query the status of the lock. All acquire and release attempts are passed through a module that simulates $M$. In principle, $M$ simulates a VoIP provider. Thus, all acquire and release attempts do not happen instantly.

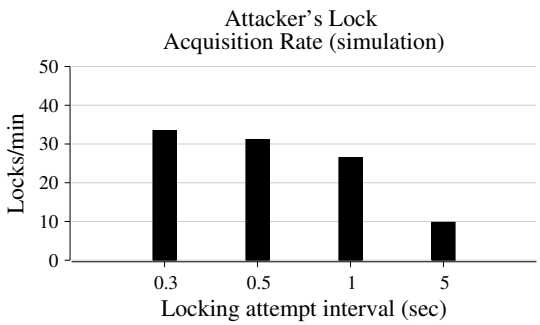


Figure 4: Rate of successful lock acquisitions managed by an aggressive thread. All attempts were issued back-to-back with a varying time interval (*locking attempt interval*) for each simulation run, ranging from 0.3 to 5 seconds.

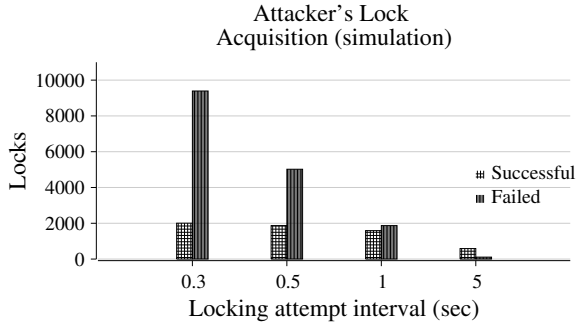


Figure 5: Distribution of all acquire attempts issued by an aggressive thread. All attempts were issued back-to-back with a varying time interval (*locking attempt interval*) for each simulation run, ranging from 0.3 to 5 seconds.

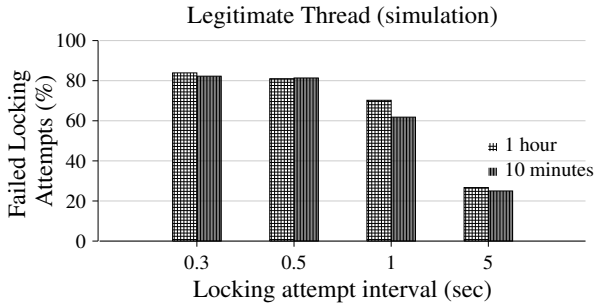


Figure 6: Percentage of failed lock acquisitions for the non aggressive thread which models a legitimate caller. All attempts were issued back-to-back with a varying time interval (*locking attempt interval*) for each simulation run, ranging from 0.3 to 5 seconds. We depict 1-hour and 10 minutes simulation runs. Observe that 10 minutes runs approximate the results of 1-hour runs.

In order to simulate the time required for an acquire operation to get fulfilled, we issued 7,300 calls through a real VoIP provider over the time period of one week (see Figure 3). 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 an acquire attempt takes place. Unfortunately, we could not follow a similar approach for the release 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 calls 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 ringing 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 release operation accordingly. Each release operation takes from 1 to 2 seconds to get fulfilled.

Based on the above configuration we issued four 1-hour simulation runs, each one having an aggressive

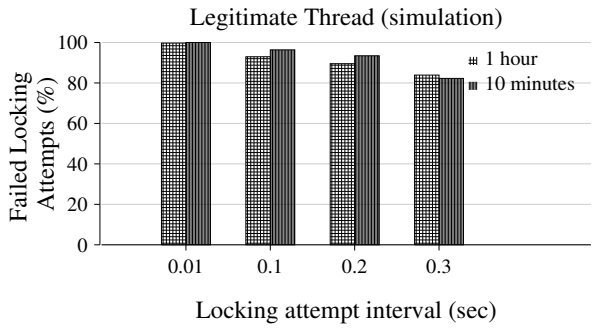


Figure 7: Percentage of failed lock acquisitions for the non aggressive thread which models a legitimate caller for extreme cases.

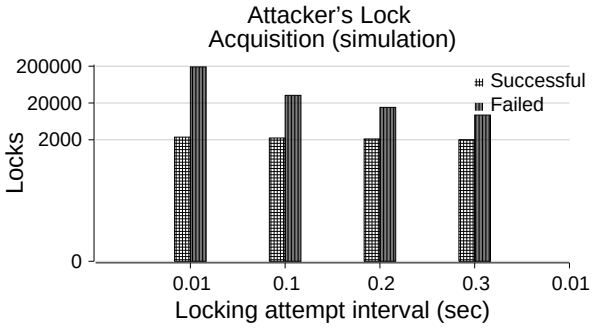


Figure 8: Distribution of all acquire attempts issued by an aggressive thread for extreme cases.

thread placing acquire operations, with different intervals. We used intervals of 0.3, 0.5, 1 and 5 seconds. We concurrently placed a thread trying to acquire the lock following a Poisson distribution with $\lambda = 10$.

We depict the rate of successful lock acquisitions an aggressive thread managed to issue in Figure 4. The best we could achieve was more than 30 acquisitions per minute. Considering that each lock acquisition implies a successful placed call, this result translates to more than 30 ringing calls per minute; a severe attack rate that would render the resource unusable.

We depict the distribution of all acquire attempts an aggressive thread managed to issue in Figure 5. Observe that as the interval reduces, the amount of failures in acquiring the lock increases. Practically there is no benefit in reducing the interval below 0.3 seconds.

We depict the percentage of successful acquire attempts the non-aggressive thread, following the Poisson distribution, managed to issue while racing with the aggressive thread, in Figure 6. First, observe that the non aggressive thread fails to acquire the lock for more than 80% of the simulation duration. Second, we can see that the first ten 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. We further discuss this issue in Section 4.

Based on the above findings we design and conduct real-world experiments in the following section.

### 3.3 Extreme Cases

It is tempting to examine the behavior of the attack for time intervals lower than 0.3 seconds. This interval basically affects the attack's impact. The probability of acquiring the lock is increased when the aggressive thread is placing acquire attempts faster. Or, in other words, having multiple concurrent acquire attempts

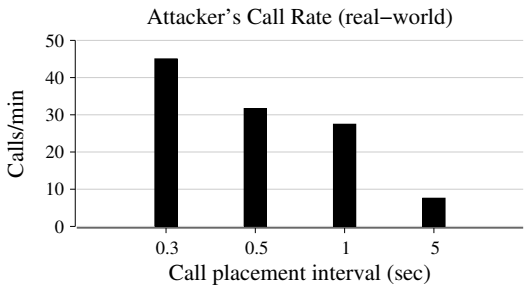


Figure 9: Rate of ringing calls managed by an adversary. All calls were issued back-to-back with a varying calling interval (*call placement interval*) for each experiment, ranging from 0.3 to 5 seconds.

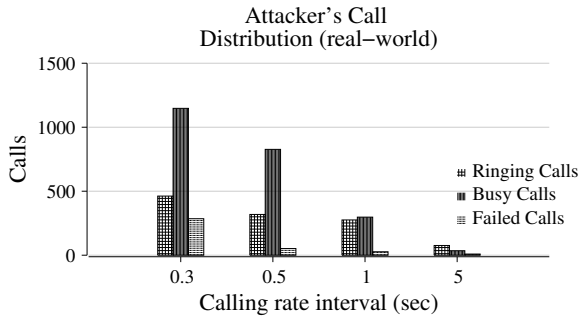


Figure 10: Distribution of all calls issued by an adversary. All calls were issued back-to-back with a varying calling interval (*call placement interval*) for each experiment, ranging from 0.3 to 5 seconds.

reduces the probability of a lock acquisition for the legitimate thread. For example, consider the case, in which acquire attempts are injected every 0.01 second. Then the legitimate thread competes with 100 lock attempts issued from the aggressive threads for the current second.

In Figure 7 we depict the impact of lower time intervals on the legitimate thread's lock acquisition attempts. Observe that for the time interval of 0.01 the aggressive thread owns the lock for 100% of the simulation run. However, this case affects negatively also the aggressive thread. Observe in Figure 8 that the majority of lock acquisition attempts issued by the aggressive thread fail. Moreover, the amount of failed attempts reach the number of about 200,000.

**Picking the ideal time interval.** As we have already discussed, the ideal interval which separates two acquire attempts in the time domain is vital for the attack. A large interval, such as 1 second, increases the probability of the legitimate thread to acquire the lock. A very low interval, such as 0.01 seconds, reduces significantly the probability of the legitimate thread to acquire the lock, but results to a vast amount of failed attempts for the aggressive thread. In a real-world experiment, each acquire attempt translates to a call placement. A very low time interval, such as 0.01 seconds, produces a huge amount of call placements, that may result in a denial of service attack against the VoIP provider. Thus, for the real-world experiments which follow in the next section we use a moderate interval, which has the optimal impact with as few as possible failed calls. Indeed, for a time interval of 0.3 seconds, the resource is seen busy from the non-aggressive thread for more than 80% of the simulation run. This is our basic intuition for carrying out real-world experiments with a time interval of 0.3 seconds.

# 4 Attack Evaluation

Based on the simulated studies we carried out in the previous section, 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, not a programming lock, but an actual telephone device.

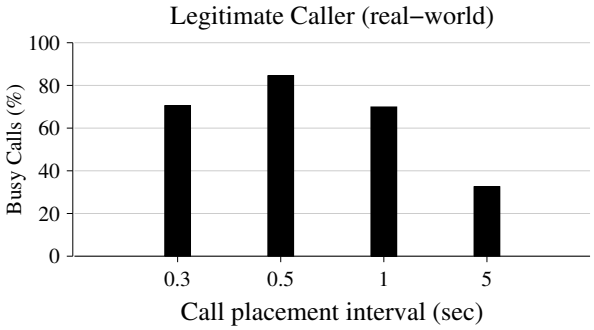


Figure 11: Percentage of busy calls received by a legitimate caller, while the target telephone device was under attack, bombarded with back-to-back calls with a varying calling interval (*call placement interval*) for each experiment, ranging from 0.3 to 5 seconds.

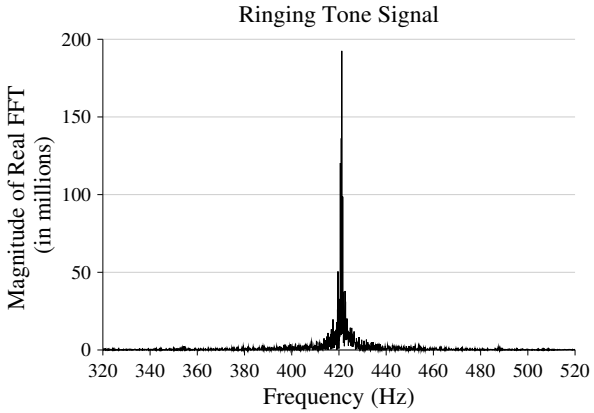


Figure 12: Ring signal frequency detection with Fast Fourier Tranformation.

### 4.1 Attack Prototype

We chose to implement the actual attack prototype using as a media a VoIP provider.[2] For the reason we chose VoIP in the first hand, please read Section 2.

The service offered by the VoIP provider we use is based on the SIP [17, 15] protocol for remote communication. SIP is the most common protocol used among different VoIP services.

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`[3] library which provides an implementation of the SIP protocol for the callers. We developed two families 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 Section 3, 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

[2]For obvious reasons we do not reveal the provider's name. However, we can provide its name upon request over private communication.

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

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. Having immediate access to the ringing status is vital for the attack. Recall from Section 2, that 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. To overcome this issue we implemented a detector module, based on a Fast Fourier Transformation of the incoming audio signal. This approach is more generic, as it is independent of the signaling protocol, in our case SIP, and always applicable, because having access to the audio signal when calling is essential. Specifically for SIP the audio stream is transported over the Real-time Transport Protocol (RTP) [16]. The audio signal of each placed call is received over RTP and is analyzed by the detector module to detect if the call is ringing. Once the detector analyzes the signal and detects a ringing tone, we place a call termination SIP message. In Figure 12 we plot the spectrum of a ringing signal, as it is identified by the detector for a real call.

### 4.2 Real-World Experiments

We conducted several real-world experiments over the period of eight months using as a victim a landline device located in our lab. 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. Recall from Section 3 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. Apparently for the case of the 1 second interval, the approximation difference is about 8%. 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 9. Recall that the attacker can detect the ringing calls by using Fast Fourier Transformation at the incoming audio signal. Observe, that the results are highly consistent with the simulated ones (see Fig. 4). The adversary has managed to issue over 45 ringing calls/minute for a calling placement interval of 0.3 seconds.

We present the distribution of all call attempts by the attacker in Figure 10. Again, the results are highly consistent with the simulated ones (see Fig. 5). Note, that as the call placement interval reduces, the fraction of busy calls increases, having a negative impact on the attack.

Finally, in Figure 11 we present the 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.

### 4.3 Discussion

Through Sections 3 and 4 we modeled, designed and evaluated experimentally a Dial attack. A threat that uses as a carrier an Internet platform, a VoIP provider, and as a victim any telephone line.

We provisioned using a simulated environment and an analytic model, that the attack may render a telephone device practically unusable. We confirmed our results using a real-world attack prototype. In fact, we managed to hold an existing landline busy for 85%, by artificially injecting short-lived synchronized calls towards it. We consider that our attack's impact has a wide range, from simple annoyance to complete disruption of critical operations. We believe that our everyday life is tightly coupled with the services offered

by the traditional telephony network. 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.

To the best of our knowledge, this is the first attempt for a systematic exploration of this new field of research. Inevitably, we left various dimensions unexplored, which may further amplify the attack's firepower or cause different effects, according to the adversary's goal. For example, consider our attack prototype driven by a distributed BotNet or consider the attack prototype incorporating multiple VoIP providers from all over the globe. No doubt, these variations may impose a larger threat, but we have a strong feeling that our results from the exploration of the attack space, as they have been outlined in this paper, will constitute the foundation on which such attacks will be based.

# 5 Existing Countermeasures

In this section we investigate existing countermeasures currently employed by three real-world VoIP providers. We do not reveal the providers' names in this paper, since publishing information collected through reverse engineering efforts is against the providers' terms of usage. Instead, we refer to each of the involved parties using the symbolic names $V_A$, $V_B$ and $V_C$, respectively.[4] $V_A$ is a proprietary service offering VoIP communication, while $V_B$ and $V_C$ are free VoIP providers based on the SIP protocol.

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. We summarize our findings in Table 1.

## 5.1 Provider $V_A$

Provider $V_A$ is considered one of the world's leading VoIP providers. This is reflected by its hundreds of millions of user accounts. More than 15 million users have been concurrently connected to the system and 300,000 simultaneous calls being served without any service degradation. As opposed to most VoIP providers that use the SIP protocol, $V_A$ relies on proprietary software and protocols that do not interoperate with SIP-based VoIP networks. We chose to focus on revealing $V_A$'s security measures against users that try to misuse the infrastructure. Our approach was to use $V_A$ to launch attacks with different configurations and test how $V_A$'s mechanisms detect such attacks.

$V_A$ 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, using $V_A$ as the VoIP carrier. 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* user of $V_A$ provider. 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.

$V_A$ maintains call history data for a period of six months. We used the call logs from the $V_A$'s web site to create the call history of each account and telephone line. In Figure 13 we present the accumulative time of the call history of each blocked account. The experiments took place over a period of about three months.

Our initial intuition is that $V_A$ blocks our account when we pass a specific call-rate threshold. Apparently,

[4] We can reveal the providers' names upon request over private communication.

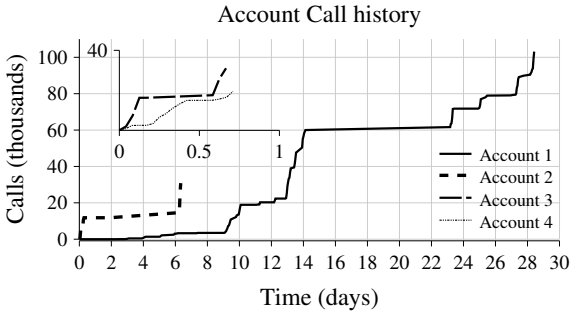


Figure 13: Call history of each $V_A$'s account, until it is blocked due to abusive behavior.

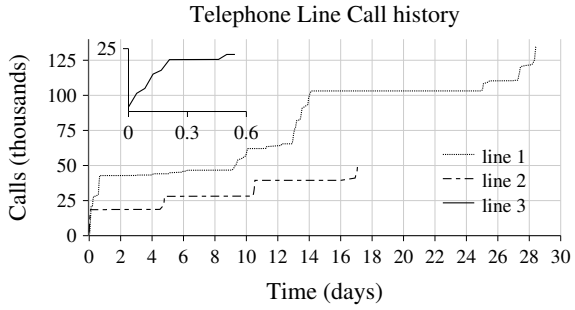


Figure 14: Call history of each telephone targeted through $V_A$, until it is blocked.

this threshold must be based on heuristics rather than being deterministic.[5] We believe this, since each account got blocked when it exceeded a totally different threshold. True enough, by using 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 13 by an almost vertical increase of at least fifteen thousand calls.

In addition, $V_A$ also blocked the landline telephone numbers which we used as victims. In Figure 14 we plot the call history of these numbers. Again, all logs are normalized to a common start time. The graphs terminate at the time the blocking actually happened. The VoIP service allowed us to place more than one hundred and thirty thousand 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 $V_A$'s anomaly detection is highly inefficient and, most importantly, it can be further abused. As far as efficiency is concerned, 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. As far as the potential abuse of the anomaly detection system is concerned, we showed that an adversary can intentionally block certain devices from $V_A$. All she needs is to issue a vast amount of missed calls towards the victim device. Note, that we didn't observe any correlation between the accounts and the victim landlines. In other words, the anomaly detection system banned the victim devices completely from the system. No one could reach these landlines through $V_A$; not only our accounts but *any user of* $V_A$.

## 5.2 Public VoIP Providers

Apart from $V_A$, which is a proprietary service offering VoIP communication, we also experimented with two public VoIP providers, $V_B$ and $V_C$.

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

On the other side we speculate that $V_C$ 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

[5] It is also possible that the anomaly detection is based on human intervention and manual inspection of call logs.

blocked the accounts used as well as all the 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 our attacks.

We summarize all our experimental findings for all the three VoIP providers we tested in Table 1.

| Provider | Description | Reaction |
|---|---|---|
| $V_A$ | Proprietary Service | Heuristic based |
| $V_B$ | Public (SIP) | No reaction |
| $V_C$ | Public (SIP) | Ban of all accounts |

Table 1: Reaction of all three real-world VoIP providers while they are used as attack platforms for Dial attacks.

# 6 Server Side Countermeasures

In Section 5 we investigated currently employed countermeasures by VoIP service providers of various sizes. Our findings concluded that existing solutions are rather inefficient. Based on our analysis, we proceed in this section and propose an anomaly detection system that promotes fairness to callers and is able to successfully mitigate the attack outlined in this paper. We chose to stay away from an *all-or-nothing* policy. In fact, our policies are dynamic and flexible. We implement our system and prove that it can dramatically reduce the attack. The adversary can hardly keep a telephone device in busy state for more than 5% of the attack duration. Recall from Section 4, that our attack prototype managed to keep a telephone device busy for 85% of the attack duration.

This section is organized as follows. We first enumerate the basic building blocks of our solution. We then proceed and present an experimental evaluation. We carry out our attack once again, but force the attacker to pass her requests through our anomaly detection system. We finally discuss our findings at the end of this section.

## 6.1 Basic Building Blocks

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[14]. 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`.[6] 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[10], 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. We present in detail these two modules, the detection and policy enforcement.

**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*[7]

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

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

| Policy | Effect | Implementation | Type |
|---|---|---|---|
| soft-mute | Drops every invite message | `iptables` (software) | *mute* |
| hard-mute | Drops every invite message | Click (hardware) | *mute* |
| hard-shape | Applies a fixed rate to invite message delivery | Click (hardware) | *shape* |

Table 2: Policies supported by our anomaly detection system. Each action is applied in messages received by a host flagged as suspicious for a period of time equal to *pew*.

requests per second. Since we have no access to the VoIP provider software we can not flag users, but only hosts (based on IP addresses). 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: (i) mute and (ii) 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. Again, we consider as legitimate behavior message arrivals following a Poisson distribution ($\lambda = 10$). It was tempting to also apply policies found in existing literature, as we reviewed in the related work section. In fact, we did try a policy for *selective dropping*, which drops packets according to the Random Early Detection (RED) algorithm[7].[8] Recall from Section 8 that 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[5, 7]. 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 provide software implementation for the *mute* action using `iptables` in Linux.[9] We provide a hypothetical hardware implementation of both *mute* and *shape* actions using the emulation environment provided by Click. Please, refer to Table 2, for a complete summary of the policies we support, along with their notation.

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

Our original intuition for *abt* was to select a value that characterizes a behavior, significantly deviating from that of a legitimate behavior. Recall from Section 3, that we have assumed that a legitimate behavior follows a Poisson distribution ($\lambda = 10$). However, in order to eliminate false positives we decided to use a more tolerable *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

[8] More precisely, we used the *gentle* version of RED[6], which is the RED implementation for Click.

[9] In a minor note, we trigger the Linux iptables mechanism from Snort using the SnortSam plugin: `http://www.snortsam.net/`.

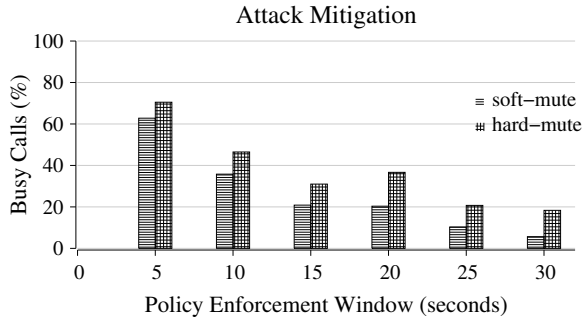


Figure 15: Attack mitigation for soft-mute and hard-mute policies for various *pew*. Notice, we do not provide results for hard-shape values for any *pew*, since the hard-shape policy is enforced for the whole attack duration. This is not explicitly forced by our detector, but it comes from the fact that the attacker does not adapt to the policy, and the *pew* is always extended.

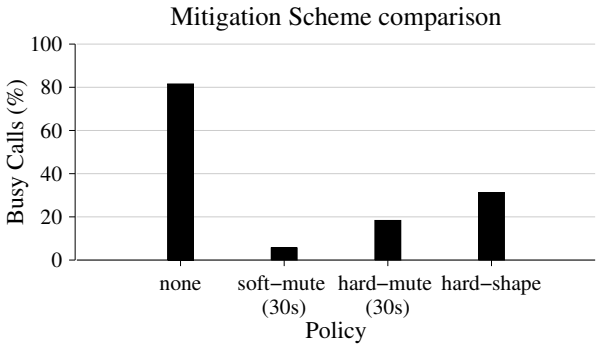


Figure 16: A comparison of all policies along with the original attack. For each policy we state the *pew* used inside parenthesis. Observe, that the attack's firepower can be reduced to 5% using *soft-mute* or down to 30% using a more relaxed policy, *hard-shape*.

in all experiments), relaxing *abt* is negative for the mitigation result. The attacker has more opportunities to bypass the legitimate behavior and still be under *abt*.

In Figure 15 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 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 16 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* policy, since the suspicious host is not muted and thus the policy is more tolerable in enforcing restraints on false positives.

# 7 Future Work

As part of our future work we consider the experimentation for purely client-side countermeasures. Our plan is to explore the usage of CAPTCHAs in telephones, since these devices have little means for defending themselves against a Dial attack. In this section we present preliminary results from a full functional call center incorporating CAPTCHAs, as well as a user study which demonstrates the applicability of our system. Note that first-time users managed to successfully solve the CAPTCHAs in 71% to 83% of the cases.

## 7.1 Call-Center Architecture

The goal of our system is to protect landlines from Dial attacks, assuming that VoIP providers have not employed any countermeasures. Our system is composed by well known software and hardware solutions which are widely available in the market. The software we use is distributed for free and the hardware costs less than 100 dollars. Below, we describe both software and hardware in detail.

**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.

**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 Asterisk was programmed to handle all incoming calls to a single landline.

**Phone CAPTCHAs.** A 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 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. This mechanism prohibits automated calls from binding to the end device and consuming resources, which could prevent legitimate callers from reaching the destination number. An attacker must incorporate automatic speech recognition software (ASR) in an 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 appropriately designed. 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.

Figure 17 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.

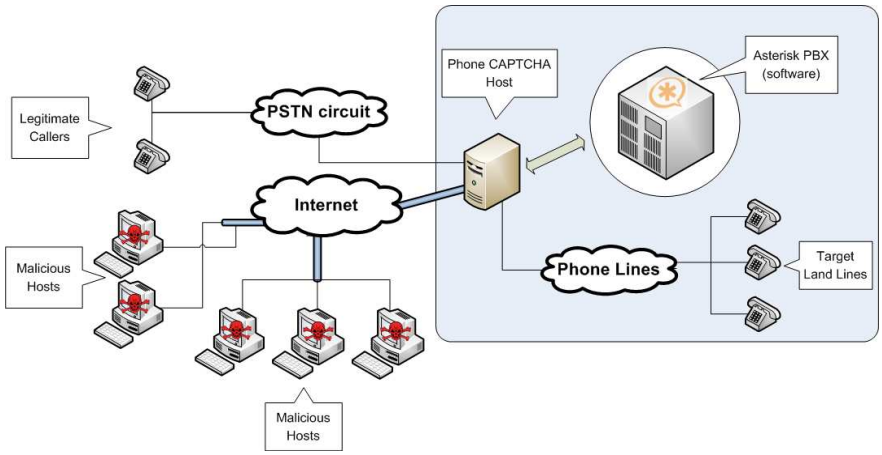


Figure 17: Diagram of a call center incorporating Phone CAPTCHA technology as a defense measure.

## 7.2 User case study

We present the results of the user case study we conducted using our proof-of-concept CAPTCHA enabled call-center. Our goal is to measure the usability of the call-center and utilize user-feedback to improve phone CAPTCHA design. The 14 test subjects participate in the study are students and staff from our

campus, between the ages of 22 and 32. They are randomly separated into two user groups, the Informed Group and the Uninformed Group. The members of the first group are fully informed of the nature of the experiment, while those of the second group are simply asked to dial a phone number. The users of the first group know that there would be a succession of 15 phone CAPTCHA tests, separated into 3 sets of tests. The first 5 tests are requesting the user to spell a word, the next 5 to type the result of a simple mathematical calculation, while the next 5 are 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 3: Success rates(%) of the user study.

In Table 3 we present the results. As expected, the informed group achieves 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.[10] In our experiment both user groups have the same success rate (71%) in the final test set. The users of both groups score 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 contain a significant amount of noise which led users to mistakes because they couldn't always make out the words. Moreover, since the Phone CAPTCHAs are in English and the test subjects have 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 solves 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 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 results in lower success rates. On the other hand, the spelling type CAPTCHAs have a much higher success rate and can utilize an immense vocabulary, making them far more robust against automated voice recognition attacks.

# 8 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. 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 [21]. 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 [23]. Zhang *et al.* in [26] exploit the reliability and trustworthiness of the billing of VoIP systems that use SIP [15]. A lot of work has also focused on man-in-the-middle attacks [24, 25] and voice pharming [22], where VoIP users are tricked and their calls do not reach their intended destinations. In this paper we explore new ways for abusing VoIP services as well as identifying possible defenses to this abuse.

[10] Previous studies suggest that blind users are able to solve 43-46% of audio CAPTCHAS deployed by popular web services, while sighted users achieve 40-50% success rate on the same tests [4]. Using sophisticated interfaces, the success rate of the same set of blind users can increase up to 70%.

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 [18, 20]. 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 [5, 7]. Enck *et al.* demonstrate the ability to deny voice service by just using a cable modem [5]. 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.

The vast amount of Internet connected mobile devices has arrised another concern. Smart-devices can be misused and launch attacks against emergency call centers [9]. In fact, cellular botnets can be formed and the large scale compromise and coordination of these mobile phones can be used to attack the core of cellular networks [19]. This approach is the opposite of ours; we use the Internet connected VoIP providers and their services to attack the telephone devices, while this approach uses the devices to attack the network. In [13] Racic *et al.* targeted their attack also against telephone devices, in the sense that they drained the battery power of internet connected mobile phones through the use of MMS vulnerabilities.

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 [12] 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. This is because emergency services base their operation on the telephony network. Aschenbruck *et al.* report that it is possible to peer VoIP calls to public service answering points (PSAP) [3]. 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 [8].

In this paper, we explore the feasibility, impact and possible mitigation of cyberattacks against telephone lines. We measure the resources needed by an attacker to launch such an attack, we propose server-side and client-side defense mechanisms and we carry out a user case study to measure the usefulness of the proposed defense mechanisms.

# 9 Conclusion

In this paper we performed an extensive exploration of a new field of research: attacks that are carried out through an Internet media and target a non-Internet infrastructure, the traditional telephony network. We refer to this activity using the term *Dial attack* or *Digitally Initiated Abuse of teLephones*.

Initially, we presented an analytic model of our attack, using simulations parameterized with values collected by real experiments. Our theoretical findings provisioned a potential threat towards telephone devices. Indeed, we implemented a prototype and carried out the attack in the wild, proving that an adversary can keep a telephone device in busy state for 85% of the attack duration. Our attack requires zero financial resources and negligible computational resources.

Considering the severity of such a threat, we proceeded and explore defenses to mitigate the impact of Dial attacks. We started, by exploring already employed countermeasures. We performed long-term experiments, over the period of 3 months, to reverse engineer anomaly detection systems used by one of the worlds leading VoIP providers and two public VoIP providers. We concluded that current VoIP infrastructures employ countermeasures based on an *all-or-nothing* approach, they react slowly to possible abuse or they offer no protection at all. We consider all these defenses highly inefficient. Thus, we presented an anomaly detection system, based on Snort and Click, that can operate entirely in software or as a combination of both software and hardware. Our system has flexible and dynamic policies and we proved that it can mitigate Dial attacks. An adversary can hardly hold a telephone device busy for 5% of the attack duration, using a VoIP provider that has deployed our solution. Having in mind that deployment in existing servers is a hard process, and clients (i.e. telephone devices) have few, if any at all, means to defend themselves against such an attack, we proceeded one step further. We performed a preliminary study of CAPTCHA usage in telephone devices, in order to alleviate our attack and implemented a fully functional call center, that incorporates *Phone CAPTCHAs* to deal with Dial attacks. We plan on extensively exploring this defense measure in future work.

The arising threat of Dial attacks has many unexplored dimensions. We believe, that this paper is the beginning of a new arms race for attacks and countermeasures in a new field of research in Computer Security.

## References


[1] IDC Predicts Almost Half a Billion Worldwide Personal IP Communications Subscribers by 2012. `http://www.idc.com/getdoc.jsp?containerId=prUS21219408`.

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

[3] 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 the Fourth IEEE International Workshop on Information Assurance, April*, pages 13–14, 2006.

[4] 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*, pages 1829–1838, New York, NY, USA, 2009. ACM.

[5] 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*, November 2005.

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

[7] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1:397–413, 1993.

[8] C. Fuchs, N. Aschenbruck, F. Leder, and P. Martini. Detecting VoIP based DoS attacks at the public safety answering point. *Proceedings of the 2008 ACM symposium on Information, Computer and Communications Security, ASIACCS 2008*, pages 148–155, 2008.

[9] C. Guo, H. Wang, and W. Zhu. Smart-phone attacks and defenses. In *Proceedings of the Third Workshop on Hot Topics in Networks (HotNets III)*, 2004.

[10] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18:263–297, 2000.

[11] P. Kretkowski. VoIP: How Free Can It Be. `http://www.voip-news.com/feature/voip-how-free-can-be-120307/`.

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

[13] R. Racic, D. Ma, and H. Chen. Exploiting MMS vulnerabilities to stealthily exhaust mobile phones battery. In *Proceedings of Security and Privacy in Communication Networks (SecureComm)*, 2006.

[14] M. Roesch. Snort - lightweight intrusion detection for networks. In *LISA '99: Proceedings of the 13th USENIX conference on System administration*, pages 229–238, Berkeley, CA, USA, 1999. USENIX Association.

[15] 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), June 2002. Updated by RFCs 3265, 3853, 4320, 4916.

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

[17] H. Schulzrinne and J. Rosenberg. Signaling for internet telephony. In *Network Protocols, 1998. Proceedings. Sixth International Conference on*, pages 298–307, 1998.

[18] 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*, October 2006.

[19] P. Traynor, M. Lin, M. Ongtang, V. Rao, T. Jaeger, T. La Porta, and P. Mcdaniel. On cellular botnets: Measuring the impact of malicious devices on a cellular network core. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS)*.

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

[21] 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*, pages 81–91, New York, NY, USA, 2005. ACM.

[22] X. Wang, R. Zhang, X. Yang, X. Jiang, and D. Wijesekera. Voice pharming attack and the trust of VoIP. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks table of contents*. ACM New York, NY, USA, 2008.

[23] 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 *Proceedings of the 16th USENIX Security Symposium*, pages 1–12, Berkeley, CA, USA, 2007. USENIX Association.

[24] R. Zhang, X. Wang, R. Farley, X. Yang, and X. Jiang. On the feasibility of launching the man-in-the-middle attacks on VoIP from remote attackers. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 61–69. ACM New York, NY, USA, 2009.

[25] R. Zhang, X. Wang, X. Yang, R. Farley, and X. Jiang. An Empirical Investigation into the Security of Phone Features in SIP-Based VoIP Systems. In *Information Security Practice and Experience: 5th International Conference, ISPEC 2009 Xi'an, China, April 13-15, 2009 Proceedings*, page 59. Springer, 2009.

[26] R. Zhang, X. Wang, X. Yang, and X. Jiang. Billing attacks on sip-based voip systems. In *Proceedings of the first USENIX workshop on Offensive Technologies (WOOT'07)*, pages 1–8, Berkeley, CA, USA, 2007. USENIX Association.