CAPTCHuring Automated (Smart)Phone Attacks

Iasonas Polakis, Georgios Kontaxis and Sotiris Ioannidis
Institute of Computer Science,
Foundation for Research and Technology Hellas, Greece
email: {polakis, kondax, sotiris}@ics.forth.gr

***Abstract*—As the Internet has entered everyday life and become tightly bound to telephony, both in the form of Voice over IP technology as well as Internet-enabled cellular devices, several attacks have emerged that target both landline and mobile devices. We present a variation of an existing attack, that exploits smartphone devices to launch a DoS attack against a telephone device by issuing a large amount of missed calls. In that light, we conduct an excessive study of *Phone CAPTCHA* usage for preventing attacks that render telephone devices unusable, and provide information on the design and implementation of our system that protects landline devices. Subsequently, we propose the integration of Phone CAPTCHAs in smartphone software as a countermeasure against a series of attacks that target such devices. We also present various enhancements to strengthen CAPTCHAs against automated attacks. Finally, we conduct a user study to measure the applicability of our enhanced Phone CAPTCHAs.**

I. Introduction

The advancement of computers has also led to the evolution of telephone technology. From traditional PSTN networks and mobile devices we have moved on to the era of Voice over IP (VoIP) and smartphones. Integration of the Internet into everyday life has led to the demand for web access on-the-go and the widespread adoption of new generation mobile devices. Such Internet-enabled devices are becoming increasingly popular, with smartphone users expected to exceed 1 billion worldwide by 2014 [8]. Additionally, VoIP subscribers will reach almost half a billion worldwide by 2012 [6]. Thus, one can expect that in the near future, legacy telephony technologies will slowly become obsolete. Nonetheless, in this transitional period where such technologies co-exist, we can expect the emergence of new threats that exploit their interconnection. As demonstrated in our previous work [22], as well as in the wild [12], an attacker can leverage VoIP technology to flood traditional telephone devices with a large number of missed calls[1] and render them unusable. We demonstrated the feasibility of such an attack, which we refer to as DIAL attacks, by leveraging VoIP technology.

Additionally, by exploiting vulnerabilities in smartphone software, attackers have proven the feasibility of issuing phone calls towards arbitrary numbers [2]. Based on these two attack classes, we argue that one can perform a DIAL attack against a target telephone device using traditional cellular networks instead of VoIP technology. This variance of DIAL attacks preserves the key characteristics of its initial form and also presents a very important advantage for the attacker; it requires zero financial resources from the attacker in all cases. Even if the victim answers an incoming call, the compromised smartphone will be charged.

[1]The attacker initiates a large number of calls which are terminated immediatelly so the victim can not answer them.

While smartphones have been targeted by a small number of malware to date, experts predict that they will attract a large number in the immediate future [7], [3], as attackers will be tempted by the built-in payment mechanism available in phones. Security vendors have already caught malware that targets smartphones, and issues a series of phone calls towards premium-rate numbers for profit.

In this work we expand the notion of Phone CAPTCHAs as a countermeasure against DIAL attacks. We explore several axes upon which they can be improved. We also propose their use as defense mechanisms against several recent attacks that target smartphones. Our key contributions are summarized as follows:

- As shown in our previous work, end telephone devices have little means to defend themselves from a DIAL attack. To mitigate this effect, we implemented a fully functional call center incorporating *Phone CAPTCHAs* for protecting telephone devices from such attacks. Furthermore, we propose a series of improvements to traditional audio CAPTCHAs to strengthen them against voice recognition attacks.
- We expand the idea of DIAL attacks and demonstrate that by exploiting a vulnerability in a smartphone, one can leverage cellular networks for flooding a target telephone device with calls.
- We propose the modification of smartphone operating system API calls to incorporate client-side Phone CAPTCHAs so as to prohibit compromised devices from issuing arbitrary calls.
- We conduct a user study that demonstrates the applicability of Phone CAPTCHAs, as first-time, non-native users managed to successfully solve the CAPTCHAs in 71% to 83% of the cases. We consider this to be very satisfactory for the newly introduced CAPTCHAs.

II. Attacks targeting landline, mobile and smart phones

In this Section we present a series of attacks that target telephone devices. Initially we review attacks that have been presented in the past, and continue with a new attack scenario that combines chracteristics of previous attacks.

A. *First generation DIAL attacks*

These attacks leverage a series of characteristics inherent in VoIP technology to target legacy telephone devices (both landline and cellular). An adversary is able to incapacitate a telephone device or calling center and obstruct legitimate callers from getting through. As we demonstrated in [22], the attack is carried out by injecting a large amount of missed calls towards a target telephone number and, thus, rendering it unusable. The adversary uses the SIP [26] protocol to register and communicate with a VoIP provider for the routing of the attack calls. By carefully placing a large number of call initiation and termination pairs, the attacker can keep the target device continuously *busy* and hinder legitimate callers from accessing it. This attack aims to disrupt the normal operation of a telephone device and can indirectly lead to profit for the attacker [12].

B. *"Smart" Dialers*

With the wide adoption of smartphones, an old familiar attack has resurfaced. Dialers [10] used to infect computers with Dial-Up Internet access and reconfigure their modem to place calls towards premium-rate numbers. DSL connections, which operate over a virtual private circuit with the ISP, had rendered such attacks ineffective. However, smartphones combine telephone devices capable of "dialing-in", with a sophisticated environment capable of executing arbitrary code and, at the same time, offer a full-featured browser access to the Internet. Therefore, smartphones present a large attack surface as their users visit arbitrary sites on the web. In late 2008, a bug [9] in Apple's Safari web browser for the iPhone device, could be exploited by a malicious website to initiate a phone call, to a destination chosen by the attacker, without user interaction. Furthermore, in 2009, a security researcher demonstrated a technique [23], [5] for discovering software vulnerabilities in smartphones and also exploiting them, all via SMS. An exploitation of such a vulnerability can result in the malware infection of the phone. It is, therefore, evident that smartphones suffer from the same flaws as standard computers and are also appealing targets because of their dial-up capabilities. They can be exploited to deliver distributed DIAL attacks or commit financial fraud with unauthorized charges. These attacks take advantage of the built-in billing system that mobile phones have, and result in direct profit for the perpetrator.

C. *Second generation DIAL attacks*

The first generation of DIAL attacks was a result of the inter-connection of traditional telephony networks and a relatively new technology, namely Voice Over IP. The second generation, emerges from the integration of a new set of capabilities, traditionaly found in computers, in mobile devices. This bundling of disjoint services allows the exploitation of one service to gain access to the other. Thus, building on the notion of DIAL attacks, an adversary can exploit vulnerabilities in smartphones to flood a target telephone device with a large amount of missed calls. Additionaly, the adversary can masquerade his attack by hiding it inside a seemingly innocent application. In both cases, the adversary instructs the smartphone device to issue calls towards an arbitrary telephone number. For this type of DIAL attacks, traditional mobile telephony networks are used as the attack medium as opposed to VoIP providers. In spite of the built-in billing system of smartphones, this attack does not aim to exploit it and lead to a monetary profit for the adversary, but rather uses it to carry out the attack.

III. Client Side Countermeasures

In this Section we present in detail the defense mechanism we implemented to protect landline devices from DIAL attacks. Telephone devices currently have no means of defence against the attack outlined in this paper. Our goal is to enable a potential target to defend against an attack utilizing IP telephony technology regardless of the countermeasures that Voip providers may, or may not, incorporate. We are, to the best of our knowledge, the first to implement a complete system that can hinder attackers from rendering a landline useless. Our solution is based on *Phone CAPTCHAs*. A CAPTCHA[30] is a challenge test that requires a response before an action can be performed, and is used in computing to ensure that 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. Although there is much debate over the use of CAPTCHAs there is plenty of recent academic effort related to this area [17], [21], [33].

A. *Architecture*

The goal of our system is to protect landlines from the DIAL attack as described in Section II. Furthermore, it can be used to block automated SPAM over IP Telephony (SPIT) calls [25], 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). Hardware private branch exchanges are used to make connections amongst the internal telephones of an organization. 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. Asterisk also supports Interactive Voice Response (IVR) technology, and can detect touch tones, i.e. 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.

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. With the use of such specialized hardware and Phone CAPTCHAs, organizations and home users can defend against VoIP-based DoS attacks targeting landlines, as described next. Figure 1 is a diagram of a call center incorporating Phone CAPTCHA technology to be deployed as a defence measure against the attack outlined in this paper. The prototype we implemented, protects a single landline, as opposed to the diagram that depicts a call center protecting numerous landlines.

B. Phone CAPTCHAs

We use Phone CAPTCHAs as a countermeasure to the attack described in our previous work. Our Phone CAPTCHA is a type of CAPTCHA crafted for use with the Asterisk PBX, but that could easily 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 amount of time to answer the CAPTCHA test using the phone's dial pad. The Phone CAPTCHA test requires the caller to press a sequence of keys based on the instructions presented to him by a recorded message. 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, which could prevent legitimate callers from reaching the destination number. Even if the attacker probes with high rates and terminates the calls immediately upon receiving a RINGING tone, our system is not affected since these calls never get past the Phone CAPTCHA to the final destination. With our defense mechanism, attackers must incorporate automatic speech recognition software in their effort to successfully launch an attack. On the other hand, it is trivial for legitimate callers to pass the phone CAPTCHA test.

A fundamental requirement for Phone CAPTCHAs to be effective against multiple attackers is the utilization of call queues. With the use of call queues, incoming calls are sent to a queue where they must pass a Phone CAPTCHA test before they are forwarded to the destination number. Without call queues, if the attackers can simultaneously issue more phone calls than the number of the target's available phone numbers, it will be nearly impossible for legitimate users to reach an available number.

The digital circuits of traditional PSTN lines are the basic granularity in telephone exchanges. That means that they have one channel and can only handle a single phone call. Higher capacity circuits such as the the T1 and E1 lines can multiplex 24 and 32 channels respectively. When a PSTN line is called, even though the call will be handled by Asterisk, and may never be forwarded to the end device, the line is occupied. Consequently, organizations with a limited number of PSTN lines cannot effectively utilize Phone CAPTCHAs against attackers with many resources, but can still deploy them as a filter against automated SPIT calls. With higher capacity circuits, multiple calls can be multiplexed through a single line and until the incoming calls match the number of available channels, the line will be available.

With the combined use of Phone CAPTCHAs and call queues, automated calls will not be forwarded to the end devices and high probing rates can be sustained. The critical infrastructure that is most likely to be targeted by attackers will be equipped with higher capacity circuits and multiple phone lines. It is common practice for organizations of this type, that have many available phone lines but only a limited number of personnel, to rely on traditional call queues to cope with multiple users. In these cases, Phone CAPTCHAs can be highly effective against this type of attack, since only legitimate callers will be forwarded to the personnel.

C. Limitations

In this Subsection we discuss a series of inherent limitations to our countermeasure platform that stem from the limitations of the system's individual components.

Attacking the infrastructure. Asterisk handles all incoming and outgoing calls and, thus, the system's effectiveness is bound by the maximum number of simultaneous

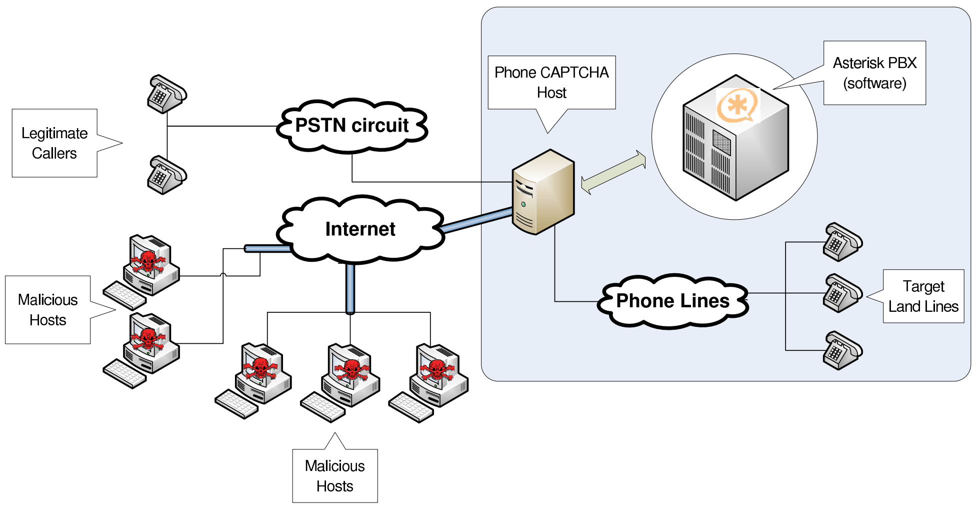

Figure 1. Diagram of a call center incorporating Phone CAPTCHA technology as a defence measure.

calls that can be handled by Asterisk, and its robustness against high calling rates. Attackers that don't have the ability to automatically solve the Phone CAPTCHAs may launch a DoS attack on the system by flooding it with such a large number of calls that Asterisk won't be able to handle. That will result in incoming calls being dropped or even the whole system crashing. For this attack to be successful, a much larger number and rate of incoming calls is demanded than in the case of an attack against a landline. Further investigation is needed to determine the threshold after which Asterisk is rendered ineffective, and whether a solution that relies on a cluster of Asterisk hosts and utilizes load balancing techniques can reinforce the infrastructure against such attacks. Even though this is a possibillity, measurements show that if Asterisk is deployed on a commodity desktop, it can easily handle more than 40 concurrent calls and can, thus, sustain the attack.

Breaking the Phone CAPTCHA. Phone CAPTCHAs are vulnerable to attacks that utilize automatic speech recognition (ASR) software to transcribe the audible information. For this type of attack [27], the adversary must first use sample data, preferably from the target speaker, to train the classifier. Even for a limited vocabulary, a large number of training samples is needed. After the training phase, the audio from the Phone CAPTCHA test can be input to these trained classifiers that will try to recognize and extract the information. Once the information has been extracted, the malicious script can send the corresponding DTMF signals to pass the test. The duration of the "deciphering" phase may vary depending on the presence of noise and distortion in the audio signal. For a successful attack, the model must decipher the message and answer the Phone CAPTCHA within the limited time before Asterisk terminates the call. In the Section V we present a series of enhancements for Phone CAPTCHAs that we implemented as well as some future directions.

Blocking legitimate callers. Even though the solution of a Phone CAPTCHA is a trivial task for a person under normal circumstances, under extreme conditions (e.g. panicked due to a fire, robbery etc.) people may not possess the mental lucidity to correctly answer the CAPTCHA. To overcome this, our defence mechanism can be utilized only when the infrastructure is receiving more calls than it can process and can use sampling to select only a number of calls to forward to the queues. That way, the phone line infrastructure can be offloaded since not all calls are presented with a Phone CAPTCHA. However, further investigation is needed to determine what types of heuristics could lead to effective sampling policies.

IV. Using Phone CAPTCHAs in smartphones

In this section we present the local use of Phone CAPTCHAs by smartphones, so as to prevent automated dial-out attacks. As mentioned in Section II-B, there have been successful exploits, misusing the phone's web browser, to dial arbitrary numbers without the user's interaction or knowledge. Example cases involve the abuse of browser URIs (e.g., tel:+123456789 or sms:+123456789, similar to http:), tapjacking [11] techniques and software (such as a valid game or application) that dials premium-rate numbers in the background [4]. Automatic dialing of arbitrary numbers could be leveraged, not only for reaching premium-rate destinations (financial fraud), but also for performing distributed DIAL attacks, similar to the ones described in Section II.

A. *Preventing Automated Phone Actions*

To prevent phone initiations we propose a modification to smartphones where local Phone CAPTCHAs are presented by the smartphone's operating system and lie in-line beneath the phone's API calls (i.e., Phone.Talk(number)). The user is presented with a CAPTCHA, which he must successfully solve for the action to continue. It is undesirable for a user-level application to be able to circumvent the CAPTCHA challenge and directly dial the number. Therefore, considering that user level applications only have access to the API calls and not directly to system calls, our modifications are limited to the programmable interface. It is necessary to add the necessary logic, to all calls providing access to phone I/O capabilities, to present a CAPTCHA challenge and block the action until it is correctly solved. We propose to use a wrapper for each sensitive API call, for instance write a wrapper sTalk (or secureTalk) for the Talk() number-dialing API call. Such a wrapper will implement the challenge logic and either call the original Talk() or not. Of course, it is necessary for the original Talk() to be withdrawn from public API access. A straightforward way to do this, and not break current phone applications, is to rename Talk() to xTalk() and change the interface scope from public to private and at the same time, rename the sTalk() wrapper to Talk().

As requiring the user to solve such puzzles every time he initiates a call is frustrating, we define the following heuristic to detect cases where such input will be required:

(a) a number is about to be dialed, which is NOT present in the "recent outgoing calls" history
(b) Phone calls are issued towards emergency services (optionally such calls may be white-listed entirely or presented with a CAPTCHA only upon several consecutive calls in a short period of time).

B. Limitations

The use of local Phone CAPTCHAs, presented by the smartphone's operating system, can be bypassed if the OS itself is compromised by malware. The attacks described above employ techniques to access the phone's dial API for calling numbers or sending SMS. However, a smartphone rootkit that lives inside the system's core has direct access to the device I/O interface. Therefore, such malware can disable the CAPTCHA mechanism or bypass it completely, in which case the operating system is unaware of any calls taking place. While vendors are trying to hinder such actions with several techniques such as the Android permission system [2], there is no, currently available, technique to detect rootkits on smart phones [14]. Of course, CAPTCHA-solving can be enforced by the server, in which case it will not be possible to circumvent. However, such an implementation requires the server to keep a history of recent outgoing calls and present CAPTCHA challenges only for previously unseen ones, in accordance to the heuristics presented above.

Furthermore, if the attacker manages to forge the phone's history file, he could inject arbitrary destination numbers which would then be dialed without the need for CAPTCHA solving. However, current API calls either offer read-only access to the phone call history or none at all.

[2]http://developer.android.com/guide/topics/security/security.html

V. 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.* [27] this type of CAPTCHAs 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 Section we propose and implement enhancements that lead to the creation of more robust phone CAPTCHAs.

Expanding the vocabulary. The effectiveness and performance of speech recognition software is greatly affected by the training phase. A large number of labeled sample data is needed during this phase to correctly train the model to recognize a specific set of words. Even for a small set of words, a very large number of training samples is needed for the system to achieve high success percentage. Traditional audio CAPTCHAs rely on a very limited vocabulary, namely digits, for security. An example 4-digit phone CAPTCHA could be: "dial-eight-one-four-three".

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. Based on that principle, phone CAPTCHAs can randomly select a word and ask the caller to spell it. [3] This will exponentially increase 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.

We intend on exploring the following dimensions for the further enhancement of phone CAPTCHAs.

Speech distortion. Removing noise from audio signals has been an active research area exhibiting successful results[19], [15], [16]. The use of algorithms to distort the speech signal itself, and not merely add background noise, in a way that would not result in CAPTCHAs unintelligible for humans, might prove to be an effective way to render speech recognition software inefficient against phone CAPTCHAs. The software would be much less accurate and efficient as it would have a high word error rate (WER) and higher real time factor (RTF). However this requires a lot of research and testing before definitive conclusions can be drawn.

Incorporating semantics. The previous enhancements aim to prolong the duration of the ASR's training phase by increasing the vocabulary, or to render the system slow and inaccurate. These enhancements result in raising the bar for attackers trying to break phone CAPTCHAs. We propose

[3]For example, the phone CAPTCHA could say "spell chair". Then, based on the letter-to-dialpad mapping, the caller would have to dial "24247".

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 not only be able to recognize words from a vast vocabulary, but also 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 very sophisticated software. By issuing more elaborate questions, that are still answerable by people, would require the attackers to have software far more advanced than what is available today.

For our proof-of-concept countermeasure implementation we have implemented three types of phone CAPTCHAS. The first type vocalizes a series of digits. This type is the most trivial to be broken since it relies on a limited vocabulary and doesn't incorporate any noise, other than that of the recording and transmission media, i.e. the microphone and phone line. The second type requires a mathematical operation on two vocalized numbers, subsequently incorporating a larger vocabulary than the previous type. The final type requires users to use the dial pad to spell a vocalized word in order to pass the test. This type is the most robust against speech recognition attacks since it can vocalize any word from a a potentially huge vocabulary. Asterisk can utilize Festival[1] to dynamically synthesize the audio files for words from a dictionary. However, this would make it easier for an attacker to break the CAPTCHA since she could easily create a large number of training data. It is safer to use pre-recorded sound files and, preferably, from various speakers to create more robust CAPTCHAs. In our implementation, Asterisk randomly selects from a pool of words pre-recorded from a single speaker.

A. 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.

In Table I 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. In our experiment both user groups had the same success rate (71%) in the final test set. The 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.

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

Table I
SUCCESS RATES(%) OF THE USER STUDY.

VI. RELATED WORK

Wang *et al.* [31] are able to identify and correlate VoIP calls anonymized by low latency networks, through a watermarking technique that makes the inter-packet timing of VoIP calls more distinctive. Wright *et al.* in [32], build a classifier that is able to identify with good probability the language used by callers, based on the VoIP packet sizes. Zhang *et al.* in [34] exploit the billing of VoIP systems that use the Session Initiation Protocol (SIP) and are able to bill users for calls that never happened or over-charge them for ones that did [26].

Research for attacks that target cellular telephony networks has been carried out in the past. Traynor *et al.* [28] argued that it is sufficient to reach a sending rate of 165 SMS messages per second, to incapacitate the GSM networks all over Manhattan. They further explore such attacks in [29]. Enck *et al.* in [18] demonstrate the feasibility of using a simple cable modem to obstruct voice service. They claim that with the use of a medium-sized botnet it is possible to target the entire United States. They also present a series of countermeasures against SMS-based attacks. These include separating the voice and data transmission channels, provisioning for higher resource utilization, and rate limiting on-air interfaces. Nauman *et al.* [24] provide a policy enforcement framework for the Android that enables users to grant selective permissions to Android applications and prohibit the use of specific resources.

The research community has also investigated the possible implication of attacks against emergency services since emergency services base their operation on the telephony network. Aschenbruck *et al.* [13] report that it is possible to peer VoIP calls to public service answering points (PSAP). This peering can have grave implications because it makes it possible to carry out DoS attacks against emergency call centers. Thus, based on the technique presented in [22] it is possible for adversaries to target and take down emergency services by flooding ther call centers with a large amount of missed calls. Countermeasures that are to be deployed by VoIP providers were presented by the authors, and can lead to the mitigation of such attacks. Fuchs *et al.* in [20] investigate the applicability of intrusion detection in emergency call centers.

VII. Conclusion

In this paper we explore the use of Phone CAPTCHAs as defense mechanisms for a series of attacks targeting telephone devices. Initially, we build a fully functional call center to protect landlines from DIAL attacks. All incoming calls are placed in queues, where they are presented with a Phone CAPTCHA puzzle that the caller must solve before the call is forwarded to the telephone device. If the caller fails to answer correctly, the call is terminated. Therefore, automated calls never get through to the device which remains available for legitimate callers.

Next, based on the vulnerabilities that exploit smartphones to dial arbitrary numbers, we expand the notion of DIAL attacks and outline a new attack. By exploiting a smartphone, an adversary is able to issue a large number of missed calls towards a target telephone device. In this attack, the adversary no longer leverages VoIP technology, but leverages the advanced capabilities of smartphone devices.

To defend against this new type of attack, as well as those seen in the wild, we propose the incorporation of Phone CAPTCHAs in smartphone operating systems. By rendering the solution of a Phone CAPTCHA puzzle a prerequisite for issuing a phone call, we can successfully hinder adversaries from carrying out their attacks. Nonetheless, issuing a puzzle for every call would have a negative impact on the smartphone usability, and therefore we present a series of heuristics to be used by the smartphone operating system to decide whether a Phone CAPTCHA must be solved before the call is issued or not.

As shown in our previous work, however, traditional audio CAPTCHAs can be easily broken. Therefore, we propose a series of enhancements that harden CAPTCHAs against voice recognition attacks. By mapping words to numbers using a phone's dialpad and incorporating semantics in tests, an adversary will require far more sophisticated software than what is available today to break our CAPTCHAs.

Finally, we conduct a user study to measure the applicability of our enhancements for Phone CAPTCHAs. Our preliminary results show that users are able to solve the puzzles in 71% to 83% of the cases. Taking into account that the test subjects were not native English speakers, we consider these results to be very promising and demonstrate that our proposed enhancements can strengthen traditional audio CAPTCHAs against adversaries without hindering legitimate users from solving them.

Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement 257007. This work was supported in part by the Marie Curie Actions – Reintegration Grants project PASS. We thank the anonymous reviewers for their valuable comments. Iasonas Polakis, Georgios Kontaxis and Sotiris Ioannidis are also with the University of Crete.

References

[1] The Festival Speech Synthesis System. http://www.cstr.ed.ac.uk/projects/festival/.

[2] About the security content of the iPhone 1.1.1 Update. http://support.apple.com/kb/HT1571.

[3] dark READING - Smartphone Malware Multiplies. http://www.darkreading.com/insiderthreat/security/attacks/showArticle.jhtml?articleID=225402185.

[4] F-Secure - Trojanised Mobile Phone Game Makes Expensive Phone Calls. http://www.f-secure.com/weblog/archives/00001930.html.

[5] Forbes - How to HiJack 'Every iPhone In The World'. http://www.forbes.com/2009/07/28/hackers-iphone-apple-technology-security-hackers.html.

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

[7] Mobile malware will increase once crooks figure how to profit from it. http://www.rcrwireless.com/article/20101001/BSS_OSS/101009990/0.

[8] Number of Smartphone Users to Quadruple. http://www.marketwire.com/press-release/Number-of-Smartphone-Users-to-Quadruple-Exceeding-1-Billion-Worldwide-by-2014-1136308.htm.

[9] SecurityFocus - iPhone Safari phone-auto-dial vulnerability . http://www.securityfocus.com/archive/1/504403/30/0/threaded.

[10] Symantec - Glossary: Dialer. http://www.symantec.com/business/security_response/glossary.jsp#d.

[11] Tapjacking: owning smartphone browsers. https://media.blackhat.com/bh-us-10/whitepapers/Bursztein_Gourdin_Rydstedt/BlackHat-USA-2010-Bursztein-Bad-Memories-wp.pdf.

[12] Thieves Flood Victims Phone With Calls to Loot Bank Accounts. http://www.wired.com/threatlevel/2010/05/telephony-dos/.

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

[14] Bickford, J., O'Hare, R., Baliga, A., Ganapathy, V., and Iftode, L. Rootkits on smart phones: attacks, implications and opportunities. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems &#38; Applications*, HotMobile '10.

[15] Deng, L., Droppo, J., and Acero, A. Recursive estimation of nonstationary noise using iterative stochastic approximation for robust speech recognition. In *Speech and Audio Processing, IEEE Transactions on* (2003).

[16] Deng, L., Droppo, J., and Acero, A. ALGONQUIN: Iterating Laplace's Method to Remove Multiple Types of Acoustic Distortion for Robust Speech Recognition. In *Speech and Audio Processing, IEEE Transactions on* (2005).

[17] Elson, J., Douceur, J., Howell, J., and Saul, J. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In *Proceedings of the 2007 ACM Conference on Computer and Communications Security (CCS).*

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

[19] Frey, B. J., Deng, L., Acero, A., and Kristjansson, T. ALGONQUIN: Iterating Laplace's Method to Remove Multiple Types of Acoustic Distortion for Robust Speech Recognition. In *7th European Conference on Speech Communication and Technology* (2001).

[20] Fuchs, C., Aschenbruck, N., Leder, F., and Martini, P. Detecting voip based dos attacks at the public safety answering point. In *ASIACCS '08: Proceedings of the 2008 ACM symposium on Information, computer and communications security.*

[21] Golle, P. Machine learning attacks against the asirra captcha. In *CCS '08: Proceedings of the 15th ACM conference on Computer and Communications Security.*

[22] Kapravelos, A., Polakis, I., Athanasopoulos, E., Ioannidis, S., and Markatos, E. P. D(e|i)aling with VoIP: Robust prevention of dial attacks. In *Proc. 15th European Symposium on Research in Computer Security (ESORICS 2010).*

[23] Mulliner, C., and Miller, C. Fuzzing the phone in your phone. In *BlackHat USA 2009* (July 2009).

[24] Nauman, M., Khan, S., and Zhang, X. Apex: extending android permission model and enforcement with user-defined runtime constraints. In *ASIACCS '10: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security.*

[25] Phithakkitnukoon, S., Dantu, R., and Baatarjav, E.-A. VoIP Security: Attacks and Solutions. *Inf. Sec. J.: A Global Perspective 17*, 3 (2008), 114–123.

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

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

[28] Traynor, P., Enck, W., McDaniel, P., and Porta, T. L. Mitigating Attacks on Open Functionality in SMS-Capable Cellular Networks. *12th Annual International Conference on Mobile Computing and Networking* (2006).

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

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

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

[32] Wright, C. V., Ballard, L., Monrose, F., and Masson, G. M. Language identification of encrypted voip traffic: Alejandra y roberto or alice and bob? In *SS'07: Proceedings of the 16th USENIX Security Symposium.*

[33] Yan, J., and El Ahmad, A. S. A low-cost attack on a microsoft captcha. In *CCS '08: Proceedings of the 15th ACM conference on Computer and Communications Security.*

[34] Zhang, R., Wang, X., Yang, X., and Jiang, X. Billing attacks on sip-based voip systems. In *WOOT '07: Proceedings of the first USENIX workshop on Offensive Technologies.*