SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks (2010)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.218

RESEARCH ARTICLE

From mobile phones to responsible devices

Patrick Traynor[1]*, Chaitrali Amrutkar[1], Vikhyath Rao[2], Trent Jaeger[2], Patrick McDaniel[2] and Thomas La Porta[2]

[1] School of Computer Science, Georgia Institute of Technology
[2] Systems and Internet Infrastructure Security Laboratory, The Pennsylvania State University

ABSTRACT

Mobile phones have evolved from simple voice terminals into highly-capable, general-purpose computing platforms. While people are becoming increasingly more dependent on such devices to perform sensitive operations, protect secret data, and be available for emergency use, it is clear that phone operating systems are not ready to become mission-critical systems. Through a pair of vulnerabilities and a simulated attack on a cellular network, we demonstrate that there are a myriad of unmanaged mechanisms on mobile phones, and that control of these mechanisms is vital to achieving reliable use. Through such vectors, mobile phones introduce a variety of new threats to their own applications and the telecommunications infrastructure itself. In this paper, we examine the requirements for providing effective mediation and access control for mobile phones. We then discuss the convergence of cellular networks with the Internet and its impact on effective resource management and quality of service. Based on these results, we argue for user devices that enable predictable behavior in a network—where their trusted computing bases can protect key applications and create predictable network impact. Copyright © 2010 John Wiley & Sons, Ltd.

KEYWORDS
cellular security; mobile phones; vulnerability analysis

***Correspondence**
Patrick Traynor, School of Computer Science, Georgia Institute of Technology.
E-mail: traynor@cc.gatech.edu

1. INTRODUCTION

Cellular networks are driving the next generation of general purpose computing platforms. The combination of always-on connectivity and device portability offer a wide range of new opportunities. From remote access to email to use as a trusted token in financial transactions to emergency response communications, researchers and developers increasingly task mobile phones with critical operations. Unfortunately, currently available device operating systems exhibit a nearly complete lack of control over access to operations and resources.

Such problems result in a radical violation of underlying design assumptions upon which telecommunications networks were built. Specifically, because applications running on mobile phones in the past were limited, the behavior of user devices and the traffic they created were largely predictable. However, emerging platforms, with malleable software configurations, cannot be depended upon to behave in any particular manner. Because device operating systems do not regulate how applications use phone resources, these devices are subject to a variety of fundamental vulnerabilities. The combination of systemic device weakness, networks with a narrow tolerance for unexpected behavior, and an ever-increasing pool of applications creates an exceedingly fragile environment incapable of providing the guarantees expected of a telecommunications network.

In this paper, we investigate the current state of security in mobile phone operating systems. This analysis provides a number of important contributions. First, we identify two novel example vulnerabilities to expose inadequacies in mediation, access control, resource management and quality of service mechanisms. Second, these vulnerabilities are then viewed in the context of their impact on larger telecommunications systems. In particular, we show how a compromised device can be used to jam communications in a single sector and characterize the effectiveness of such an attack against battery lifetime. Finally, we show that while previous work in systems security research provides some guidance, mobile phones introduce a variety of new ways for attackers to exploit applications and the network infrastructure itself. We examine the requirements on the phone system mechanisms necessary to control such threats. We also argue for a philosophical change, where phones must be made *smart* about their impact on resource usage, both about the effect of their operations on network service behavior and about how to reduce resource costs to themselves, in order to remain as mission-critical devices. By providing these devices with more capable and reliable trusted computing bases, we greatly increase their ability to protect key applications and create predictable network impact.

2. VULNERABILITY ANALYSIS

In order to provide specific examples of mechanisms in mobile phones, we examine the Symbian operating system [25]. While a number of other prominent alternatives including variants of Linux and Windows exist, we have selected Symbian because it represents more than 50% of the smart phone market share [5]. While we focus on the specific details of Symbian, we note that other related platforms including the iPhone and Android are expected to be increasingly targeted [16,11]. Moreover, the recommendations that we will make are platform agnostic and can benefit all mobile devices.

Symbian is based on a microkernel architecture, supporting only memory management and scheduling within the kernel itself. All remaining components of the base subsystem, including process management and the filesystem, are provided via modules in user space. Additional subsystems providing telephony, graphics, security and an application framework sit above the base subsystem. Figure 1 offers an overview of this system.

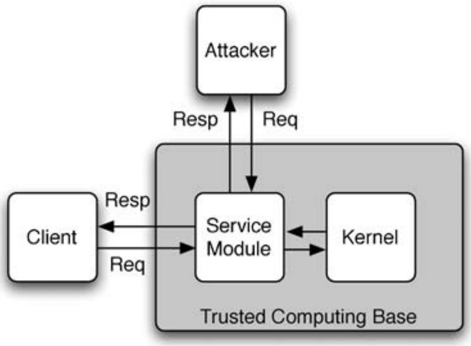

Figure 1. The architecture of the Symbian operating system. Because there is only a single user, requests by an attacker are treated the same as those made by the legitimate client.

In keeping with its status as the most widespread OS, Symbian is also the most attacked [14]. However, other mobile device operating systems have historically contained similar serious weaknesses [12,18]. Vulnerabilities including DLL overwriting, weaknesses in Bluetooth security, capability circumvention, and an increasing pool of malware (Cabir [7], Mabir [8], Skulls [9], Commwarrior [10] etc) represent a small sampling of currently available exploits. The specific examples below are novel in their illustration of new and systemic weaknesses in mobile phone operating systems, but achieve similar ends as previously published vulnerabilities. Note that these exploits are possible using standard API calls available to all applications.

2.1. Keylogging

The Window Session Server controls all GUI-based programs running on a Symbian-based device. In order for applications to respond to user input, this service returns specific key press events to each application calling the CaptureKey() function. For example, a game may call CaptureKey() for all directional arrows on the keypad. As we discovered on Symbian 60 and Symbian 80 systems, this function returns key presses to all requesters, regardless of the application in focus. Because each application can call CaptureKey() for multiple keys, we were able to implement a program running in the background capable of logging all buttons pressed by the user.

Keylogging is actually enhanced by other artifacts of mobile phones. On each button press, for example, mobile phones emit a *Dual-Tone Multi-Frequency* (DTMF) tone. Each DTMF tone uniquely identifies the button pressed such that a receiver can decode and determine the associated numerical value. The uses of this technique are widespread and include resolving user PINs. However, such tones can also be *locally* decoded by an adversary to recover the same information. By simply recording a small portion of a phone call, triggered by a button push after its initiation, an adversary could log keys aurally.

The dangers of keylogging on such a platform are becoming increasingly acute. Service providers are already beginning to aggressively market mobile phone access to bank accounts and bill payment [23]. Users are also regularly asked to enter sensitive information such as credit card and social security numbers when dealing with customer service lines.

2.2. Remote command execution

Attention (AT) commands are used to send instructions to a modem. Operations including establishing voice calls, sending SMS messages, and accessing phonebook data can be executed using such messages. In previous generations of mobile phones, the use of AT commands was restricted through a serial cable for testing and troubleshooting purposes. However, the widespread use of Bluetooth technology allows AT commands to be issued wirelessly via the Dial Up Networking (DUN) profile. Using weaknesses in Bluetooth allowing an adversary to add themselves to a connection white-list, an attacker can remotely send commands to nearby phones. Because the DUN service does not perform additional access control checks, the attached adversary can perform actions including sending voice/data calls and SMS messages without the permission or knowledge of the affected host.

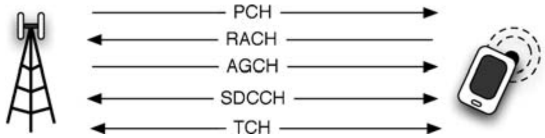

Figure 2. A high-level description of the GSM air interface. Phones are paged on the PCH, respond on the shared RACH, are instructed to listen to a dedicated control channel on the AGCH, authenticate, perform maintenance routines and receive text messages (where applicable) on the SDCCH and engage in conversations on the TCH.

While the remote execution of AT commands was previously accomplished by BlueBug [34], this previous exploit takes advantage of another service—the serial port profile. By demonstrating the ability to recreate such vulnerabilities across multiple services, we reinforce our assertion of the lack of control being an endemic property of these operating systems.

2.3. Network impact

With a high degree of accuracy, voice call duration, frequency and even destination have become predictable parameters in telecommunications networks. Accordingly, such systems have evolved and been optimized around anticipated user behavior. The level to which these networks are tailored to expected behavior is easily observable through their reaction to non-standard traffic patterns. For example, the advent of dial-up modems forced service providers to vastly increase capacity as users began occupying call circuits for significantly longer than when only voice communication was possible. Increased end device capability must therefore be examined in terms of its potential impact on the network.

Our previous work demonstrates a malicious incarnation of this problem. By generating a small volume of targeted SMS traffic, an adversary can abuse the delicate balance of bandwidth apportioning between signaling and communications channels to virtually shut down a cellular network [29,30]. While these previous attacks were launched from the Internet, they could just as easily be accomplished through the manipulation of AT commands on a large number of malicious smart phones.

2.3.1. Jamming attack.

Due to their lack of basic security mechanisms such as memory protection and the value of data stored in them, we assume that mobile devices will increasingly be targeted by rootkit-installing malware. To better understand how a compromised device can affect the network, we characterize the impact of software designed to launch a jamming attack. We begin by explaining how the air interface is used in GSM. When an incoming call or text message arrives at a base station, the tower broadcasts an alert message addressed to the phone on the *Paging Channel* (PCH). The phone, which may have its radio turned off to save power, will eventually wake up and hear the page from the network. The phone indicates its readiness to receive an incoming session by transmitting a response on the *Random Access Channel* (RACH). The RACH implements Slotted Aloha as a means of addressing channel contention. When the base station eventually receives the phone's response, it sends a message to the phone via the *Access Grant Channel* (AGCH), which instructs the phone to tune itself to a specific dedicated control channel. This channel, known as the *Standalone Dedicated Control Channel* (SDCCH), is used to perform authentication, negotiate session keys, perform maintenance routines and deliver text messages (if indeed that is the incoming session). If the incoming session is instead a phone call, the network then instructs the phone to listen to a specific *Traffic Channel* (TCH), on which the actual conversation will take place. All of the above channels are superimposed over timeslots as GSM relies on TDMA. Figure 2 offers an overview of this process.

Using a phone as a jamming device can prove unexpectedly challenging if an adversary is unfamiliar with the details of the air interface. In particular, an attacker could potentially modify a phone to interfere with a single carrier, or frequency capable of supporting up to eight concurrent calls (i.e., a carrier can have as many as eight TCHs). However, the vast majority of towers support many carriers, meaning that such an attack would be limited in its scope. Moreover, CDMA-based networks would not be impacted by such an attack as the jamming traffic created by the adversary would not affect the base station's ability to properly demodulate other calls. A more successful attack against either network would instead target the control channels (i.e., PCH, RACH, AGCH, SDCCH). By blocking the channels used to establish calls, an adversary can affect a far greater percentage of traffic in their immediate vicinity.

We focus our attack efforts specifically on the RACH. To completely jam communications, an infected phone would simply transmit messages in all of the timeslots allocated to the RACH. Because phones can already transmit legitimate responses in any timeslot on this channel (it is the only shared uplink control channel), developing malware designed to transmit in many of the timeslots would less difficult than attempting to alter the device to transmit in any of the downlink control channels. Figure 3 shows how the RACH channel is implemented. Malware attacking the RACH would need to reflash the firmware running on the baseband processor, which is directly responsible for radio operations. As this technique is being used in practice (e.g., to unlock iPhones) and can increasingly be initiated by software instead of a series of button presses [15], we believe it to be entirely within an adversary's capabilities. We therefore evaluate the impact of such behavior on legitimate network traffic.

To model this tradeoff, we modify our GSM air interface simulator [30,32] to include a RACH attack mode of operation. We note that our simulator was built according to 3GPP standards and tested to tightly conform to the prescribed behavior of real systems. The parameters of this simulator were set using information from a variety of publicly vetted

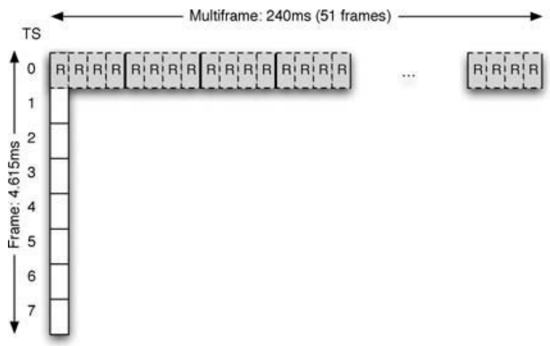

Figure 3. The creation of channels over timeslots in GSM. This figure depicts the uplink portion of the spectrum and demonstrates that all 51 timeslots are reserved for the RACH (in timeslot 0). The channels created by timeslots 1–7 would be used as TCHs.

sources. We model normal behavior in Manhattan, in which 50,000 calls and nearly 200,000 text messages are delivered over 55 sectors (transmission areas) per hour based on a Poisson random distribution. We note that this relatively heavy-looking traffic load exhibits no blocking. The means by which these parameters were chosen are discussed in the Appendix.

Figure 4 shows the impact of a compromised phone transmitting in a varying number of RACH timeslots. We vary the number of timeslots between malicious transmissions to profile the blocking characteristics of such an attack. While transmitting in every timeslot allocated to the RACH prevents all communications in a sector from being received, an infected phone may run the risk of depleting its battery. In particular, assuming that the transmission of a jamming message costs roughly the same as a voice message, a compromised phone exhibiting such behavior would be able to

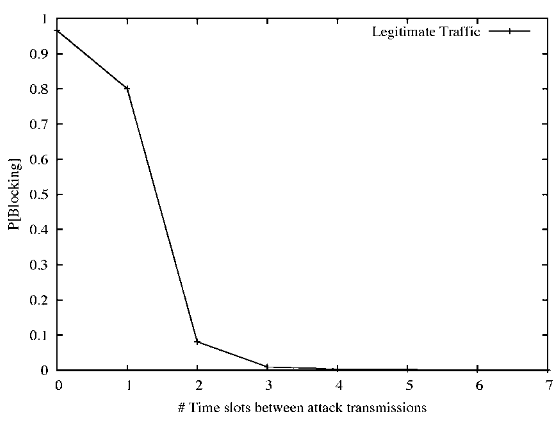

Figure 4. The impact of RACH jamming on normal traffic by varying the number of slots blocked by a malicious node. Even without transmitting during every possible frame, the use of Slotted Aloha on the RACH makes channel efficiency low during congested periods. Note that by transmitting in every other slot, an adversary can still cause 80% blocking and double its effective attack time to 10 hours.

continue its attack for the average "talk-time" of five hours.† However, an adversary may also significantly degrade traffic and cause a blocking rate of over 80% for twice this time period simply by transmitting a jamming message in every other timeslot.

Such an attack represents a significant threat to the network. For a system in which a 1% blocking rate is considered to be an operational failure, the ability to deny service to the majority of users in an area for a number of hours represents a total shutdown of the network. Unlike an equivalent attack in a wired network, where an administrator can easily locate and shut down a malicious device, recourse is extremely limited in this environment. In particular, because devices do not authenticate with the network until they are assigned an SDCCH, a service provider would be unlikely to be able to identify and disable an infected phone. The ease with which such devices can be controlled by malicious software must therefore be of great concern to service providers.

2.3.2. Other attacks.

Damage to the network is also not limited to denial of service. With significantly expanded capabilities and open functionality, devices may also probe critical network servers for weaknesses. The possibility of gaining control of once-isolated core nodes through vulnerabilities such as buffer overflows, including the Home Location Register (HLR) and Mobile Switching Centers (MSCs), becomes a greater reality given the potential to create and send targeted malformed control messages into the network. Such control could be used for activities ranging from illicit eavesdropping to fraudulent billing.

3. SYSTEMS SECURITY REQUIREMENTS

Operating systems designers have explored mechanisms for mediation, access control, resource management and quality of service in great detail over the years. However, the problems in Section 2 show that the mechanisms in current mobile phones are not sufficient. In this section, we argue that these problems are not solved in current operating systems either, and outline the requirements for providing the necessary controls for mobile phones. We use examples from the Symbian operating system and a version of Linux implementing mandatory access controls to illustrate the pervasiveness of such problems and lack of solutions in current systems.

3.1. Limitations of systems security

The vulnerabilities above demonstrate two key problems: (1) mobile phones do not effectively control access to their security-critical operations, such as the service module

† Battery lifetimes for phones vary widely depending on a number of inputs. We selected five hours as a conservative estimate.

DOI: 10.1002/sec

operations in Symbian (see Figure 1). As shown in the keylogger case, the Symbian windowing system permits any client process to access keystrokes from other processes. In the remote execution case, the Telephony Application Programming Interface ("TAPI server") does not distinguish between remote and local commands. In the presence of malicious or poorly written applications and compromised devices, (2) new threats to the telecommunications infrastructure are enabled by the lack of such protections. For example, the ability to force core network elements to repeatedly perform expensive operations can be achieved by transmitting the correct AT commands. Attacks on specific components of the network infrastructure were previously limited due to the inability to send commands to these elements from external networks.

The emergence of Mandatory Access Control (MAC) systems would appear to be an appropriate solution for the problems above. A MAC system mediates all "security-sensitive" operations in order to ensure that a system security policy is enforced [4]. Implemented on a mobile phone, such a system could help mediate access to security-critical operations. Once all operations are mediated, the operating system could limit the use of such operations by untrusted applications.

Unfortunately, MAC operating systems, such as Security-Enhanced Linux [20] (SELinux), are currently not capable of securing mobile phones against these challenges either. First, mediation in SELinux is insufficient. Linux phones [19] provide user-level services for processing AT commands and keyboard events in a fashion similar to Symbian. In the case of the former, applications submit AT commands to the TAPI server, which forwards these requests to the modem device files. Just as AT commands were executed without question on devices running Symbian, so too are such commands in Linux phones. For the keylogger example, keyboard events on Linux phones are processed by an poorly mediated user-level server. While work has been ongoing to mediate X server commands using SELinux policies [17], no implementation is currently available.

Second, MAC operating systems are not conscious of network resource usage resulting from AT commands. Specifically, mobile phones can use AT commands to request services from core elements in the telecommunication network. Whereas computers calling a mobile phone through landlines are limited in the functions they can execute (i.e., calls), mobile phones can invoke a variety of operations in the HLR and MSCs. Such functions include registration and deregistration for voice and data services, location updates, call requests, and features including call waiting, voice mail, and call forwarding. Because MAC systems such as SELinux do not enforce resource limitations, additional policy and mechanisms will be necessary to protect these telecommunications servers.

3.2. Access control

Appropriate access control for mobile phones must mediate all AT commands, windowing commands, and file access, and enforce policies that protect the system's trusted computing base from compromise. SELinux MAC enforces file access and work is underway to control X server operations,‡ so we consider how to fill in the missing pieces, controlling the AT commands and developing mobile phone policies.

While applications generally submit AT commands to the TAPI server, such commands can also be sent directly to the modem device files. AT commands must therefore be mediated not only at the TAPI server but also at the file system. In order to mediate requests made to the modem files, we propose to extend the TAPI to mediate access (e.g., using the SELinux user-level policy server [33], as the X server does). SELinux will be used to restrict modem file access to the TAPI server only.

Operating system designers have long struggled over the trade-off between access control policies that enforce fine-grained control (e.g., *least privilege* [24]) and coarse-grained permissions that are easier to manage. The Symbian systems use a coarser-grained policy that aims to protect the integrity of the system and Symbian-approved applications. The Symbian model defines three types of principals: system, Symbian-signed [26], and others. System principals, such as the installer, have complete system access. Symbian-signed processes run programs that have been authorized by Symbian. These programs cannot install new programs, but they do have additional permissions (e.g., to write signed application files). The idea is distinct from Microsoft's authenticode in that Symbian testers evaluate the code before generating a digital signature. Such a process is not foolproof, however, as programs containing spyware have gained Symbian's approval [22]. All other processes belong to the third type. While these processes cannot do as much as the system or Symbian-signed code,§ they have significant privileges; they can read all files, modify some system information (e.g., Bluetooth pairing), and make phone calls.

Some emerging mobile phone operating systems provide fine-grained access control based on MAC systems. Motorola has MotoAC which is based on SELinux [21], and others are working on similar systems. While MAC enforcement is required to protect the mobile phone's trusted computing base and key applications (e.g., software installation), we strive to achieve the simplicity of a model based on a few principals rather than complex SELinux policies. In contrast, Symbian's current "three principal" model is clearly too coarse grained. Our challenge is to develop user

‡ Although work on controlling window manager operations has not begun.

§ In Symbian 9.1, third-party applications are given access to all functions available to Symbian-signed processes. However, users must click "yes" for such calls to be successful. As no user can be expected to make effective access control decisions when inundated with a series of checkboxes, such protections are ineffective.

devices such that coarse-grained access is feasible (e.g., a principal per application class).

3.3. Resource management

The vulnerabilities examined in Section 2.3 illustrate problems in the mobile phone protecting its resources from use by external adversaries and the protection of network resources from the phone applications. Preventing such problems requires some form of resource accounting which can be based on the mediation required for access control, but the problem is often determining what resource allowances are permissible.

The circuit switched architecture of telecommunications networks is designed to provide consistent service guarantees. When used only to support real-time voice communications, the assurances provided by such a system help to ensure a uniform guaranteed quality of service to all users. However, as such systems evolve to carry both voice and data traffic, the reliance upon traditional reservation mechanisms results in a significant under-utilization of network resource. Accordingly, telecommunications networks are transitioning to support packet switched services (e.g. General Packet Radio Service (GPRS)). In order to create an environment capable of supporting service requirements ranging from best effort to real-time streaming data, both the network and user devices will need to implement and control quality of service mechanisms.

Quality of Service (QoS) has certainly been discussed from the perspective of the network [1]. In reality, however, little to no mechanisms are deployed in the current generation of telecommunications networks [6]. While such services will eventually be accessible to mobile phones in next generation (3G) networks, APIs capable of expressing flow service constraints are already available to Symbian developers. Unfortunately, such interfaces provide the associated devices with little context for making QoS decisions. For example, there is no means of limiting the QoS parameters for specific applications. Rather, any application labeling its sockets with a certain priority receives that service if it is available from the network. While the network itself is able to downgrade QoS parameters during the lifetime of a connection, the operating system itself possesses no such mechanism. Although many applications will legitimately require higher priority delivery, the ability to arbitrarily escalate an application's bandwidth privileges should generally be viewed as unsafe to both the network and the mobile phone.

4. SmartER PHONES

The title *smart phone* is a misnomer. Instead, we have created a device with vastly expanded functionality but with no additional context in which to make decisions. The vulnerabilities presented in this paper demonstrate systemic weaknesses in design, allowing for simple malware to not only easily gain access to the operations of a single user, but also damage the availability of the network for large numbers of other users. From our experience over the past few years [29,30,32,27,31,28], we argue that the problems facing this field can be divided into three distinct areas. First, device operating systems do not control their operations and resource usage. Second, because telecommunications networks have been over-engineered, even slight changes in mobile phone behavior will have an impact on network behavior. Finally, the increasing diversity of applications and open connectivity to the Internet, both of which lack context specific to cellular systems, will continue to test the limits of both devices and networks.

In short, the continued success of these networks is dependent on the creation of *smarter*, more responsible phones—devices capable of intelligently operating in conjunction with telecommunications networks, not independently of them. This recommendation includes not only making developers aware of the fundamentally different qualities of cellular networks, but also providing well controlled interfaces through which such software can operate in a predictable manner. We use a network-aware packet scheduler running as a module in a mobile phone as an example. Because every connection to the network requires significant initial overhead (registration, channel assignment, etc), establishing connections requires an expensive set of operations for both the network and mobile phones. However, this cost could be amortized by scheduling background events on mobile phones (e.g. checking for email, updating sports scores) to coincide with network initiated connections such as periodic location updates. Such traffic can be piggybacked on the same session, benefitting the mobile phone by minimizing the power and bandwidth resources spent on connection overhead. This behavior would also be beneficial for the network, by increasing the predictability of network access by the device. Such requests would occur with enough regularity[‖] without altering the perceived experience of even non-mobile users.

Accomplishing the above requires effective versions of the mechanisms discussed in this paper. Through the combination of mediation, access control, resource management and quality of service, we can begin to address much of the Byzantine and unexpected behavior previously discussed. While some of the mechanism options are known, the mobile phones and telecommunications networks have unique requirements that will require the creation of new conceptual approaches. Specifically, concerns regarding computational cost, power consumption and memory requirements must all be addressed for a mechanism to be successful in this space. Most critically, operating systems designers will need to possess a more intimate understanding of the telecommunications infrastructure of which they are a part.

[‖] The frequency of updates is provider dependent; however, anywhere between six and ten messages per hour is well within reason.

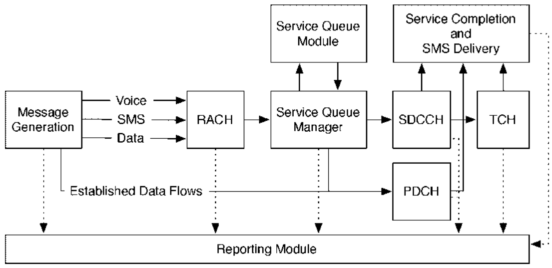

Figure 5. Simulator architecture.

The accuracy of simulation was measured in two ways. The components used by voice and SMS were previously verified using a comparison of baseline simulation against calculated blocking and utilization rates. With 95% confidence, values fell within ± 0.006 (on a scale of 0.0 to 1.0) of the mean. The simple nature of the PDCH module allowed verification of correctness through baseline simulations and observation.

5. CONCLUSION

In this paper, we have demonstrated that mobile phone operating systems currently lack the mechanisms to adequately protect these increasingly capable devices. As a result, an adversary may be able to not only cause numerous violations of a user's data confidentiality and device integrity, but also cause significant problems for the cellular networks themselves. While known techniques will help improve security in this setting, we believe that a fundamental shift in philosophy will be required to protect these mission critical systems.

APPENDIX

We extend the GSM simulator built in our previous work [30] to provide support for GPRS data service. In total, the project contains nearly 10,000 lines of code (an addition of approximately 2,000 lines) and supporting scripts. A high-level overview of the components is shown in Figure 5, where solid and broken lines indicate message and reporting flows, respectively. Traffic is created according to a Poisson random distribution through a Mersenne Twister Pseudo Random Number Generator [13], saved to a file and then loaded at runtime. The path taken by individual requests depends on the flow type.

After creation, messages proceed to the *RACH stage*, which strictly follows 3GPP TS 04.18 [2] and is tunable using the *max_retrans* and *tx_integer* variables. The *Service Queue Manager stage* assigns messages to an SDCCH. If desired, a pluggable *Service Queue Module* can be defined using standard interface callback functions. If possible, the Service Queue Manager assigns a message to an SDCCH. Rather than simulating exact communication and compensating for retransmission, messages are held in the *SDCCH stage* for an exponential mean service time corresponding to message type. For accuracy, each SDCCH services messages by decreasing counters only during frames defined in 3GPP TS 05.01 [3]. When counters reach zero, SMS messages complete and voice messages attempt to acquire a TCH. Like the SDCCH stage, the *TCH stage* uses an exponential mean hold time to simulate channel occupancy. TCHs service messages during every frame, and when the hold time counter reaches zero, the call is complete, and the TCH is released.

REFERENCES

1. 3rd Generation Partenership Project. Quality of Service (QoS) concept and architecture. Technical Report 3GPP TS 23.107 v6.4.0.
2. 3rd Generation Partnership Project. Physical layer on the radio path; General description. Technical Report 3GPP TS 04.18 v8.26.0.
3. 3rd Generation Partnership Project. Physical layer on the radio path; General description. Technical Report 3GPP TS 05.01 v8.9.0.
4. Anderson JP. Computer security technology planning study. Technical Report ESD-TR-73-51, The Mitre Corporation, Air Force Electronic Systems Division, Hanscom AFB, Badford, MA, 1972.
5. Canalys. Smart mobile device market growth remains steady at 55%. http://www.canalys.com/pr/2006/r2006071.htm, 2006.
6. Chakravorty R, Cartwright J, Pratt I. Practical Experience with TCP over GPRS. In *Proceedings of IEEE GLOBECOM*, 2002.
7. F-Secure Corporation. F-Secure Computer Virus Descriptions: Cabir. http://www.f-secure.com/v-descs/cabir.shtml, December 2004.
8. F-Secure Corporation. F-Secure Computer Virus Descriptions: Mabir.A. http://www.f-secure.com/v-descs/mabir.shtml, April 2005.
9. F-Secure Corporation. F-Secure Computer Virus Descriptions: Skulls.A. http://www.f-secure.com/v-descs/skulls.shtml, January 2005.
10. F-Secure Corporation. F-Secure Malware Information Pages: Worm:SymbOS/Commwarrior. http://www.f-secure.com/v-descs/commwarrior.shtml, 2008.
11. Ganapati P. Malware Sneaks into Android Market. http://www.wired.com/gadgetlab/2010/01/android-malware-fears/, 2010.
12. Guo C, Wang HJ, Zhu W. Smart Phone Attacks and Defenses. In *Proceedings of Third ACM Workshop on Hot Topics in Networks (HotNets-III)*, 2004.
13. Hedden J.Math::Random::MT::Auto—Auto-seeded Mersenne Twister PRNGs. http://search.cpan.org/~jdhedden/Math-Random-MT-Auto-5.01/lib/Math/Random/MT/Auto.pm. Version 5.01.

14. Hickey AR. Virus onslaught sickens smartphones. http://searchmobilecomputing.techtarget.com/originalContent/0,289142,sid40_gci1185083,00.html, 2006.
15. iphone-elite. How to Downgrade screwed up baseband 4.0 (after anySIM and 1.1.1 firmware upgrade). http://code.google.com/p/iphone-elite/wiki/DowngradingBaseband, 2007.
16. Keizer G. New iPhone worm steals online banking codes, builds botnet. http://www.computerworld.com/s/article/9141354/New_iPhone_worm_steals_online_banking_codes_builds_botnet, 2009.
17. Kilpatrick D, Salamon W, Vance C. Securing the X Window system with SELinux. Technical Report 03-006, NAI Labs, March 2003.
18. Kingpin and Mudge. Security Analysis of the Palm Operating System and its Weaknesses Against Malicious Code Threats. In *Proceedings of the 10th USENIX Security Symposium*, 2001.
19. Motorola. opensource.motorola.com. http://opensource.motorola.com.
20. National Security Agency. Security Enhanced Linux. http://www.nsa.gov/selinux/.
21. OpenEZX. Rokr e2—openezx. http://wiki.openezx.org/Rokr_E2.
22. Portal SN. Just because its signed doesnt mean it isnt spying on you. http://www.securitynewsportal.com/securitynews/article.php?title=Just_because_its_Signed_doesnt_mean_it_isnt_spying_on_you, 2007.
23. Reuters. Cingular plans mobile banking service for 2007. http://today.reuters.com/news/articleinvesting.aspx?view=CN&storyID=2006-11-15T050108Z_01_N14348236_RTRIDST_0_TELECOMS-CINGULAR-BANKING.XML&rpc=66&type=qcna, 2006.
24. Saltzer J, Schroeder M. The Protection of Information in Computer Systems. In *Proceedings of the ACM Symposium on Operating System Principles (SOSP)*, 1973.
25. Symbian Limited. Symbian OS—the mobile operating system. http://www.symbian.com/, 2006.
26. Symbian Limited. Symbian Signed. http://www.symbiansigned.com, 2006.
27. Traynor P. Characterizing the Limitations of Third-Party EAS Over Cellular Text Messaging Services. Technical report, 3G Americas Whitepaper, 2008.
28. Traynor P. Securing Cellular Infrastructure: Challenges and Opportunities. *IEEE Security & Privacy Magazine* 2009; **7**(4).
29. Traynor P, Enck W, McDaniel P, La Porta T. Exploiting Open Functionality in SMS-Capable Cellular Networks. *Journal of Computer Security (JCS)* 2008; **16**(6): 713–742.
30. Traynor P, Enck W, McDaniel P, La Porta T. Mitigating Attacks On Open Functionality in SMS-Capable Cellular Networks. *IEEE/ACM Transactions on Networking (TON)* 2009; **17**(1).
31. Traynor P, Lin M, Ongtang M, *et al*. On Cellular Botnets: Measuring the Impact of Malicious Devices on a Cellular Network Core. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2009.
32. Traynor P, McDaniel P, La Porta T. On Attack Causality in Internet-Connected Cellular Networks. In *Proceedings of the USENIX Security Symposium (SECURITY)*, 2007.
33. Security enhanced linux policy management framework. urlhttp://sepolicy-server.sourceforge.net/.
34. Trifinite. BlueBug. http://trifinite.org/trifinite_stuff_bluebug.html, 2004.