On Cellular Botnets: Measuring the Impact of Malicious Devices on a Cellular Network Core

Patrick Traynor
Georgia Tech Information Security Center (GTISC)
School of Computer Science
Georgia Institute of Technology
Atlanta, GA, 30332
traynor@cc.gatech.edu

Michael Lin, Machigar Ongtang, Vikhyath Rao, Trent Jaeger, Patrick McDaniel and Thomas La Porta
Systems and Internet Infrastructure Security Laboratory
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
{molin, ongtang, vrao, jaeger, mcdaniel, tlp}@cse.psu.edu

ABSTRACT

The vast expansion of interconnectivity with the Internet and the rapid evolution of highly-capable but largely insecure mobile devices threatens cellular networks. In this paper, we characterize the impact of the large scale compromise and coordination of mobile phones in attacks against the core of these networks. Through a combination of measurement, simulation and analysis, we demonstrate the ability of a botnet composed of as few as 11,750 compromised mobile phones to degrade service to area-code sized regions by 93%. As such attacks are accomplished through the execution of network service requests and not a constant stream of phone calls, users are unlikely to be aware of their occurrence. We then investigate a number of significant network bottlenecks, their impact on the density of compromised nodes per base station and how they can be avoided. We conclude by discussing a number of countermeasures that may help to partially mitigate the threats posed by such attacks.

1. INTRODUCTION

Attacks against the core infrastructure of the Internet have become commonplace. Services including DNS [18, 45, 12] and BGP [8, 41, 33, 46] are now regularly targeted by highly capable adversaries. While malicious behavior has caused significant problems for specific services and sets of domains, such attacks have not yet resulted in an Internet-wide collapse. Systems with more rigid hierarchical dependencies, notably cellular networks, are less likely to withstand similar maliciousness. While attacks on these networks are beginning to become more diverse, academic research in this area has only focused on two general thrusts. The first explores how the lack of authentication for signaling traffic in the wired network would allow an adversary with physical access to cause significant outages [35, 26, 13, 32]. The second investigates attacks targeting the wireless portion of the network, and results in either the saturation of a wireless link [60, 59, 43, 58] or the conversion of a mobile phone into spam generators or nodes used to attack Internet-based resources [22, 19, 39]. *However, previous studies have not investigated whether compromised mobile phones can generate sufficient amounts of traffic to impair the network core itself, yet alone attempted to measure the damage caused by such traffic.*

In this paper, we evaluate the potential for and impact of the large scale infection and coordination of mobile phones in attacks against the core of a cellular network. In particular, we characterize Denial of Service (DoS) attacks using selected service request types on the *Home Location Register* (HLR), the central repository of user location and profile information in the network, by a botnet composed entirely of mobile phones. As the HLR must be queried in the delivery of all phone calls and text messages, acts as the authentication service for the network, records data for the purposes of billing and assists in wide range of other jobs, such an attack would potentially devastate nearly all service in the network. Our results show that botnets as small as 11,750 phones can cause a reduction of throughput of more than 90% to area-code sized regions supported by most currently deployed systems. More capable databases experience a throughput reduction of approximately 75% when faced with attacks by 141,000 compromised devices. Note that while such numbers may appear large, they typically represent less than 2% and 15% of phones assigned to a single HLR, respectively. We then explore the bottlenecks that make the realization of these attacks difficult and how they can be overcome.

While such attacks may seem similar to other attacks on core infrastructure, noteworthy differences in the architecture of cellular networks make the consequences of such behavior significantly worse. Whereas widespread caching and the existence of multiple routes would help mitigate the loss of a DNS root or BGP node, respectively, the loss of an HLR would make all users assigned to this device immediately unreachable.

In this work, we make the following contributions:

- **Attack Characterization and Quantification:** We use a combination of an industry standard benchmarking tool and testing on a live network to characterize a flooding attack on an HLR. Such testing allows us to not only determine if compromised phones can generate a sufficient amount of traffic to cause large-scale outages, but also provides the first estimates of the necessary size of cellular botnets. *To our knowledge, this work represents the first investigation of such attacks on the core of a cellular network.*

- **Reduce Adversary's Workload:** Through the above tests, we are able to determine the classes of operations that force the network to do the most work, thereby reducing the effort required by an adversary seeking to cause outages.

- **Provide Intelligent Control Mechanisms:** We discuss a number of different command and control structures that will help coordinate distributed attacks and provide a means of avoiding network bottlenecks. We show that without careful planning, these attacks are likely to fail.

Most critically, this work demonstrates that access to once unaddressable components in the core creates the potential for increasingly sophisticated attacks against these networks. In this particular case, we show that the reliance upon state information used to support mobility and a variety of network services makes these systems highly susceptible to attack.

The remainder of this paper is organized as follows: Section 2 provides an overview of related research; Section 3 presents the architecture of the SS7 network core and examines a number of methods by which the large scale compromise of mobile devices could occur; Section 4 details the attack itself; Section 5 characterizes and quantifies the impact of different types of traffic on the HLR; Section 6 measures the performance of individual commands against a live network and aligns our TM1 simulations with deployed standards; Section 7 simulates an attack on the HLR; Section 8 discusses issues of cellular botnet management including command and control; Section 9 provides a number of short-term mitigation techniques; Section 10 provides concluding remarks.

2. RELATED WORK

Denial of service attacks have been studied in a wide variety of systems. Targets including DNS roots [18, 45], software vendors [24, 27], news services [47], search engines [44] and online casinos [7] have all been overwhelmed by malicious traffic. Such attacks have even impacted resources and processes in the physical world, and caused significant outages in areas such as banking, emergency and even postal services [38, 10]. The research community has responded with significant attempts to both categorize [37] and mitigate [49, 28, 29, 62, 63, 56, 55, 30, 64] such problems. Unfortunately, such attacks are only beginning to be understood in the context of cellular networks.

Previous research on the malicious overload of cellular networks focused *entirely* on the wireless domain. The first such work, by Traynor et al [59], demonstrated that a small volume of text messages targeted at a geographic area could be used to deny legitimate voice and SMS service. Serror et al. [50] showed that similar results were possible by overloading paging services, as did Lee et al. [34] on shared uplink channels. Racic et al. [43] investigated problems with resource allocation algorithms on the air interface. While it is possible to mitigate the impact of such attacks using a variety of queue and resource management techniques [58], Traynor et al [60] noted that it was the architecture of the network itself that was responsible for enabling such attacks. By attacking high-bandwidth cellular data services with less traffic than in their previous work, this work illustrated the fundamental tension caused by the meeting of best effort IP networks and the circuit-switched air interface of cellular networks. However, such attacks were previously considered to have originated from outside of the targeted network.

The transformation of mobile devices from simple voice terminals into highly-capable, general purpose computing platforms makes the possibility of attacks originating from within the network a reality. Mudge and Kingpin were among the first to discuss the insecurity of such devices in early handheld devices [31]. In their examination of PalmOS, this work encountered a lack of even the most basic operating system security mechanisms. Since that time, a number of others have further investigated a variety of vulnerable device interfaces including Bluetooth [61, 9] and the multimedia messaging subsystem (MMS) [39]. Guo et al. [22] argued more generally that connecting such unprotected "smart" devices to telecommunications may prove troublesome, but discussed only attacks equivalent to local wireless jamming, denial of service on call centers (e.g., endpoints) and spamming. While malware targeting such devices has surfaced, including Cabir [14], Mabir [15] and Skulls [16], widespread infection like that seen on the Internet has yet to be reported. Given that such devices continue to lack sufficient protection mechanisms, their ability to impact larger network functionality must be investigated.

3. OVERVIEW OF CELLULAR SYSTEMS

In this section, we provide an overview of important background information, including the architecture of cellular networks, the hardware of mobile phones and a discussion of how such devices are likely to become infected.

3.1 Network Architecture and Components

The heart of a cellular network is the *Home Location Register* (HLR). Users are assigned to specific HLRs based on their phone numbers. When someone attempts to call a user, the caller's switch queries the appropriate HLR with a request for the targeted user's current location. A variety of other services, ranging from authentication and call-forwarding to billing rely on assistance from the HLR. Accordingly, this central repository of user profile data is crucial in providing service in a network with mobile endpoints.

Mobile Switching Centers (MSCs) act as telephony switches and deliver circuit-switched traffic in a GSM network. Such a succinct description, however, fails to capture the magnitude of functions tasked to these devices. MSCs are responsible for performing "handoffs" between base stations, assist in billing operations and can function as gateways to both wired and neighboring cellular systems. With the assistance of a *Visiting Location Register* (VLR), MSCs can identify and store information about currently associated subscribers. However, information requests for other subscribers still require interaction with the HLR. For data connections, these operations are performed by the *Serving GPRS Support Node* (SGSN).

Mobile phones and other cellular-enabled devices connect wirelessly to the network through the *Base Station Subsystem* (BSS). These units provide the logic for operations such as wireless resource management, encryption and frequency hopping.

3.2 Mobile Phone Architecture

Mobile phones operate using two processors. The *Application Processor* supports functionality including traditional operating system duties (e.g., memory management) and user services such as the camera and music players. When a process needs to use the network, the Application Processor passes an Attention, or *AT command* [51], to the *Baseband Processor*. Such commands can be used to establish telephony and data links or invoke network supported services such as call waiting and registration.

AT commands can be passed to the Baseband Processor in a number of other ways. Bluetooth connections, serial links and even other applications with the necessary privileges can all inject such requests. Malware infecting such devices can therefore easily initiate interaction with the network core.

3.3 Infecting Phones

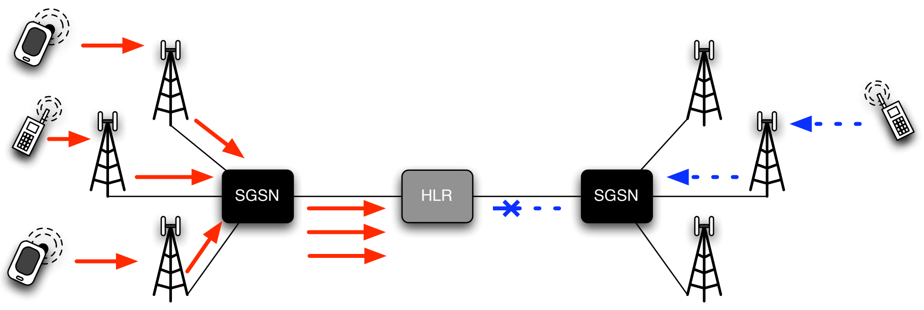

Figure 1: A high level overview of an attack on the HLR. By overloading this logically central component of the network (left), the network will be unable to service legitimate traffic (right) for very large geographic regions.

Mobile devices have rapidly transformed from limited embedded systems to highly capable general purpose computing platforms. While such devices have long enjoyed significant diversity in hardware and operating systems, the rising popularity of smartphones and the ability to sell applications to users is leading to the establishment of standardized mobile software platforms and operating systems, such as Microsoft's Windows Mobile, Google's Android and Apple's Mobile OS X. Unfortunately, many devices are only now beginning to implement basic security mechanisms including memory protection and separation of privilege. Accordingly, such systems expected to be increasingly targeted by malware.

Malware targeting mobile devices may come from any one of a number of sources. Given that 10% of cellular users downloaded games to their mobile devices at least once a month in 2007 [40][1] and the wide availability of free ringtones, downloadable content and executables are one of the more likely origins. Like their desktop counterparts, mobile devices are also likely to be susceptible to a range of browser exploits including drive-by downloads [42]. Finally, the presence of multiple communications interfaces makes mobile devices susceptible to malware that propagates not only through the cellular network itself [19], but also potentially through WiFi and Bluetooth [14, 15, 16, 17]. Accordingly, the breadth of infection vectors exceeds that of many traditional networked systems.

4. ATTACK OVERVIEW

The attack in this paper attempts to prevent legitimate users of a cellular network from sending or receiving calls or text messages. Because most of the functionality of these networks relies on the availablity and proper functioning of HLRs, our attack seeks to violate such conditions. In particular, an adversary with control over a set of compromised mobile phones will attempt to overwhelm a specific HLR with a large volume of traffic. Legitimate users relying on the same HLR will be unable to receive service as their requests will be dropped. Figure 1 shows how such an attack would occur.

There are a number of differences that make such DoS attacks different from those observed on the Internet. First, mobile devices can not transmit entirely arbitrary requests to the HLR; rather, only specific types of messages can be exchanged. Second, such requests must be made in a manner such that unnecessary traffic or side effects are not generated. Malware with auto-dialer or auto-texter functionality would not only use significantly more resources within the network (thereby increasing the obviousness of the attack to both users and administrators), but would also potentially cause serious congestion locally. This behavior may prevent such an attack from acheiving the widespread outages possible if the HLR were instead reached.

To understand such attacks, we first characterize the performance of an HLR (Section 5) and then reconcile these experiments with the behavior of a live network (Section 6). When estimating the size of a cellular botnet, we compare the relative impact of a variety of interactions with the HLR, thereby allowing an adversary to execute such an attack with the smallest amount of traffic possible (Section 7). Finally, we explore network bottlenecks and discuss ways in which they can be avoided (Section 8). We argue a real attack is more likely to occur against multiple HLRs instead of a single device as we assume that infected phones will occur uniformly with population density. However, by characterizing the behavior of a single HLR, we gain context to discuss realistic scenarios.

Table 1: TM1's Transaction Mix

Transaction	Type	Default Mix
`get_subscriber_data`	r	35%
`get_new_destination`	r	10%
`get_access_data`	r	35%
`update_subscriber_data`	w	2%
`update_location`	w	14%
`insert_call_forwarding`	rw	2%
`delete_call_forwarding`	rw	2%

5. CHARACTERIZING HLR PERFORMANCE

Because most service requests in a cellular network must pass through an HLR, these devices represent an attractive target for an adversary. In particular, the failure of an HLR can cause all users serviced by this database to be denied service. The expanded capabilities of mobile phones and their significant susceptibility to infection make the exploitation of such a choke point a realistic threat. In this section, we quantify the performance impact caused by the invocation of the range of services provided by an HLR. We achieve such characterization through the use of the Telecom One (TM1) benchmarking suite [52]. This telecommunications industry standard for database load testing allows providers to evaluate different hardware and software configurations prior to deploying equipment. Through the use of this tool, we can identify the most expensive classes of service requests, *thereby reducing the effort needed for an adversary to effectively render an HLR unavailable.*

[1]Given the popularity of application download services provided on platforms such as the iPhone and Android, such numbers are expected to grow significantly.

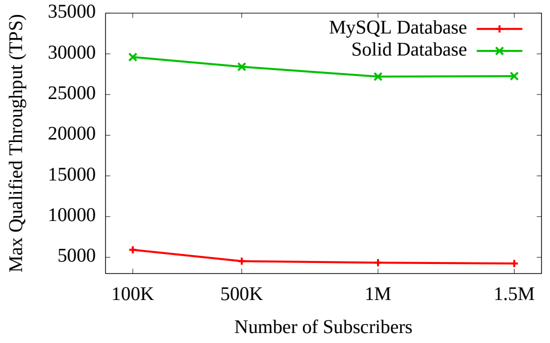

Figure 2: Throughput for an HLR with varying numbers of subscribers using TM1's Default Mix.

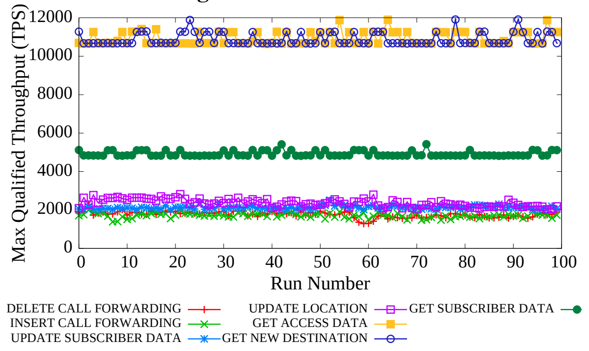

Figure 3: HLR throughput for each transaction type with 500K subscribers.

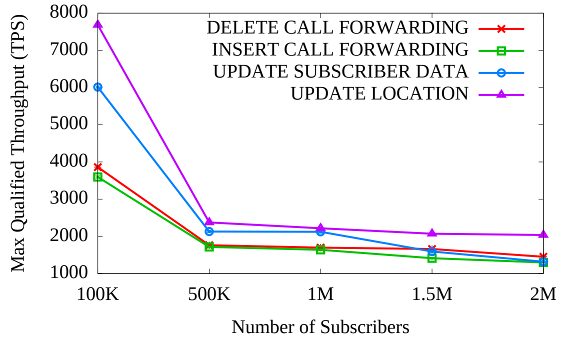

Figure 4: Comparison of candidate transaction types based on the resulting throughput in a MySQL database.

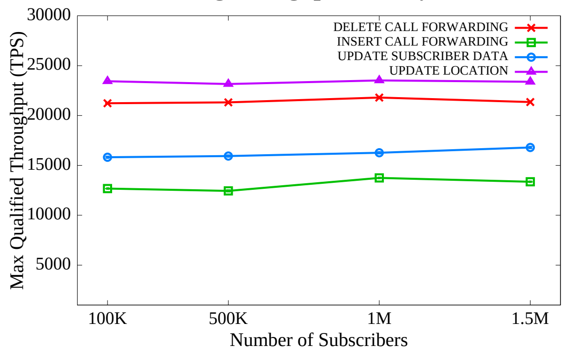

Figure 5: Comparison of candidate transaction types based on the resulting throughput in a SolidDB database.

5.1 Testbed Configuration

We used version 3.0.6 of the TM1 benchmarking suite to characterize an HLR under normal and attack conditions. TM1 creates a database and then uses a set of networked clients to generate traffic to determine the Maximum Qualified Throughput (MQTh)[2] in transactions per second (TPS). The HLR was deployed on server-class hardware with dual quad-core (8 total cores) Xeon 2.3 GHz CPUs and 8 GB of RAM running a 2.6.22 Linux Kernel. Such a hardware configuration is within [54] or above [5, 23] specification for a number of real HLRs. We use this setup to evaluate the performance of MySQL version 5.0.45 and SolidDB version 6.0, as both databases are used in many real deployments. MySQL was configured to use the InnoDB storage engine, which maintains a buffer pool for caching data and indexes in main memory. SolidDB stores the entire database in memory; thus it offers significantly higher performance. These configurations are representative of the range of HLRs that are currently (MySQL) and will be increasingly deployed (SolidDB) in the coming years. In order to simulate maximal load on the HLR, client processes were spawned across as many as six different machines, each with the same hardware specifications as the HLR. Client machines were connected to the HLR via Gigabit switch. All experiments using TM1 were repeated until the 95% confidence interval was less than two orders of magnitude of the mean (approximately 50 experiments per data point).

5.2 Normal HLR Behavior

Providers consider two factors when deploying an HLR. The first, the number of subscribers serviced by the database, is influenced by practical concerns including population density, equipment capability and available resources. Real deployments range from a few hundred thousand to five million subscribers per HLR [6]. The second factor is the rate and type of service requests from mobile devices. As discussed earlier, mobile devices interact with the HLR in well established patterns based on functions of device mobility, call rates and the anticipated frequency of specific services being activated.

TM1 allows users to simulate normal traffic by providing a "Default Mix" of read and write operations. In spite of the great variety of AT commands invoking services from mobile phones, only a small set of generic back-end operations are required to support all possible requests in an HLR. Recognizing this, TM1 categorizes all such requests into one of seven "meta-commands"[3]. As shown in Table 1, the Default Mix is comprised of 80% read and 20% write meta-commands. This mix is used as the standard comparison between systems using TM1 [23, 54, 5]. Given that users more frequently perform read operations (e.g., make phone calls, send text messages) than write operations (e.g., authenticate to the network, update their location), such a mix is consistent with reality.

Figure 2 shows the impact of the number of subscribers on the MQTh for both the HLRs running MySQL and SolidDB. Because only the caching data and indexes of the MySQL database are stored in memory, these systems are less robust in handling elevated traffic from large populations of users. Because of the high demand on the HLR, system throughput quickly becomes a result of disk throughput. The HLR running SolidDB does not suffer from the

[2]Different documents produced by SolidDB have different definitions for MQTh [52, 53]. Through experimentation, we have determined that this value actually represents the mean throughput.

[3]This term is our own. These are similar to the GSM MAP commands [3], however, there is not a direct mapping

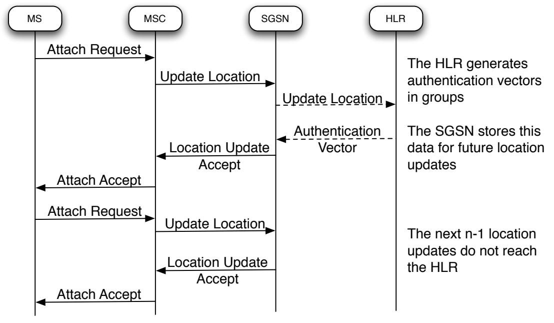

Figure 6: Simplified message flow for update location. This figure illustrates the caching of authentication vectors in the VLR and vastly simplifies the remainder of the message flow.

same significant degradation of performance because the database itself is stored in main memory and it employs a more advanced concurrency control technique. These results are consistent with previously published studies given equivalent hardware configurations [54].

5.3 Characterizing Meta-Commands

Understanding the capacity of an HLR using normal traffic mixes provides an approximate volume of traffic an attacker must inject to disrupt the network. However, the amount of attack traffic can be reduced significantly through a simple observation. An adversary recognizing that read and write operations on a database have different costs can greatly improve the efficiency of their attack. Accordingly, we characterize the performance of an HLR for each of the meta-commands.

Figure 3 shows the MQTh for each of the seven meta-commands on an HLR running a MySQL database with 500K subscribers. As expected, read-only transactions perform significantly better than those performing writes. For instance, `get_access_data` and `get_new_destination` can be processed at approximately 11,000 TPS, whereas all four of the transactions performing writes achieve less than 2,500 TPS. Even `get_subscriber_data`, which performs a large number of reads, experiences approximately double the MQTh of the fastest write transaction.

Figures 4 and 5 characterize the performance of each meta-command for a varying number of users on HLRs running MySQL and SolidDB, respectively. We omit the results of the less expensive read commands as injecting them would require the attacker to create more traffic than would be necessary by simply increasing traffic with normal characteristics. Both Figures provide a relatively stable ranking of the relative expense required to service these requests.

Through this analysis, we have demonstrated that an adversary selecting certain subsets of service requests can improve the efficiency of their attack. However, more information on the behavior of the core network is required before a successful attack can be launched.

6. PROFILING NETWORK BEHAVIOR

Not surprisingly, the experiments in the last section demonstrated that service requests requiring database writes are significantly more expensive than those performing reads. However, the magnitude of this difference, nearly six times in some cases, was not previously appreciated from an attack perspective in this environment. While such experiments certainly give insight into the classes of messages an adversary would select in order to improve the effectiveness of their attack, they do not consider the impact of network architecture and protocols. To reconcile the differences between our simulations and reality, we further characterize the impact of the write-based service requests on a live network. We then consult standards documents to verify discrepancies in expected and observed behavior.

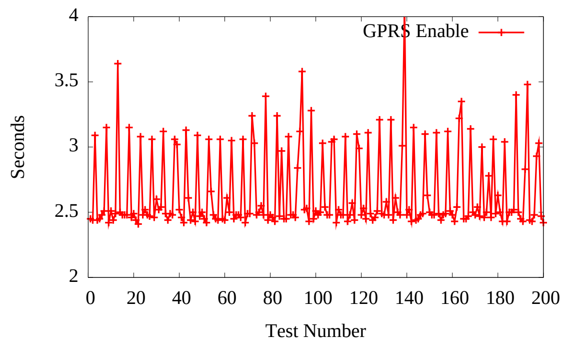

Figure 7: Response times for location updates via the GPRS attach AT command. Note that the caching of authentication vectors occurs sets of five.

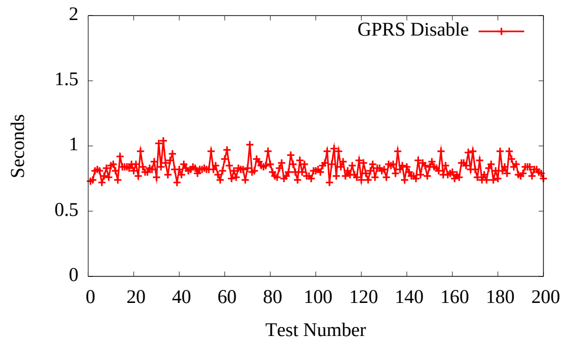

Figure 8: Response times for location updates via the GPRS Detach AT command. Note that the rapid response from the network indicates that no information is sent to the HLR.

6.1 TM1 and Measuring Network Behavior

We injected and measured service requests representing each of the four write-based meta-commands on a live cellular network. All tests used a *single* Nokia 9500 running Symbian Series 80. Similar results were recorded using a Motorola A1200 running a 2.4.20 Linux kernel. Each AT command was executed a total of 200 times during low traffic hours. In addition to the time required to receive a response from the network for each AT command, we had to insert a two second delay between the the sequential execution of commands as immediate execution on some phones caused extended delays. We therefore build in this extra time across all platforms to be as conservative as possible in our calculations. Note that our experiments were run using the utmost caution and at no time did we attempt to overwhelm the HLR.

6.2 Update Location

One of the most important functions of an HLR is to keep track of each user's location. To support such operations, mobile devices implement a series of AT commands that alert the network of any of a number of changes. For instance, as a device moves between two base stations connected to different SGSNs or MSCs, a location update occurs. The same operation occurs when a device turns

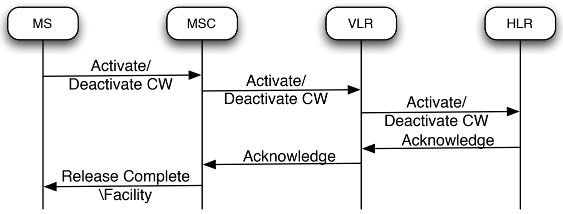

Figure 9: The messaging flow for Call Waiting requests. Note that updates always reach the HLR. The VLR stores a copy of the forwarded number in case the device attempts to later query the network.

on or off ("attaches" or "detaches") to or from the network. Finally, location updates occur periodically to ensure that a device has not improperly detached from the network (e.g., battery failure). All location update operations require that the mobile device reauthenticate itself with the network.

Location updates are well understood as expensive operations. Accordingly, the network designers developed caching mechanisms in order to minimize the impact on the HLR. The HLR amortizes the cost of device authentication by pushing most of the load to SGSNs[4]. Specifically, when an `update_location` request does reach the HLR, it responds by generating n authentication vectors [2]. The SGSN uses these vectors to perform this and the next $n-1$ authentication requests. If the location update does not change the device's SGSN, no messages are transmitted back to the HLR. Figure 6 provides a high level overview of this procedure.

Figure 7 shows the impact of such caching on a real network. In particular, we observed the response times for the GPRS Attach AT command, which connects a device to the data services provided by the network. These experiments show two clear groups of response times - clusters of four fast responses taking an average of 2.5 seconds (σ=0.2) and individual peaks lasting approximately 3 seconds (σ=0.13). Given the relatively high consistency of response time within these two groups, it is clear that this particular network caches five authentication vectors at the SGSN (i.e., $n = 5$). Such behavior makes attacks using the `update_location` meta-command difficult. In particular, because only one in five requests from an infected phone actually reaches the HLR, an adversary would need to create a significantly larger network of infected phones than for the other write-based meta-commands. Worse still, because infected devices within close proximity (i.e., the same city) would be more likely to strain their SGSN, many of these messages would never reach the HLR at all.

A companion function, GPRS Detach, offers little additional assistance to an adversary. Attempting to perform GPRS Attach fails without first deregistering from the network. Attach and detach commands were therefore interleaved for this set of experiments. Even with a far smaller response time, as shown in Figure 8, this command almost never results in the HLR receiving traffic. In fact, standards documents note that an SGSN need not update the HLR or delete authentication vectors when a device detaches from the network [2]. The low response times for the detach operation, in conjunction with the caching observed in the previous experiment, indicate that this suggestion is indeed implemented in reality.

With a total turnaround time of about 5 seconds between GPRS Attach commands (3 second response time + 2 second command

[4]This task is performed by the MSC instead in the rare cases that no SGSN is available or the mobile device does not support data service.

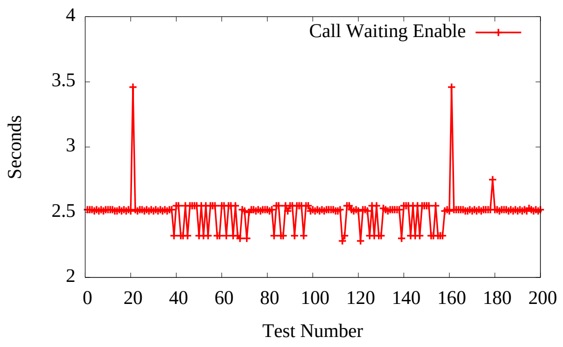

Figure 10: Response times for CW enable requests.

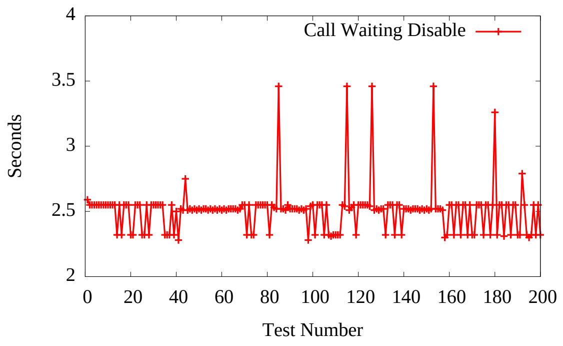

Figure 11: Response times for CW disable requests.

interval), GPRS Attach has a output rate of roughly 0.2 commands per second. However, as only one in five attach commands reach the HLR, the throughput is further reduced by one-fifth, to 0.04 commands per second.

6.3 Update Subscriber Data

There are a number of operations that require the network to update user profiles. For instance, the activation, deactivation or modification of the parameters of many services requires the HLR to modify a number of fields within the database. While these operations support a diverse set of functionality, all of their interactions with the HLR can be described as an update to two tables. Services including Call Waiting, Call Barring and Modify PDP Context all fall under this category. Although we limit our discussion to Call Waiting, it should be noted that experiments conducted on Call Barring exhibited identical performance characteristics.

Figure 9 shows the flow of messages for a Call Waiting Enable request. Unlike `update_location`, all Call Waiting enable requests are directed to the HLR. The HLR acknowledges the receipt of this request to both the client and the client's VLR, which caches whether or not call waiting is enabled for each client. This cached information is only used if the mobile phone queries the network to determine whether call waiting is currently activated. The same operations occur when Call Waiting Disable requests are sent.

Figures 10 and 11 show response times for Call Waiting Enable and Disable requests. The response times are consistent, with activation averaging 2.5 seconds for both functions with $\sigma_{enable} = 0.13$ and $\sigma_{disable} = 0.18$. Given that both Call Waiting Enable and Disable are both implemented by the `update_subscriber_data` meta-commands, such results are expected. Of interest, however, are the erratically appearing spikes in both graphs. Such deviations indicate contention for a shared wireless resource known as the *Random Access Channel* (RACH). Techniques for minimizing such contention are discussed in Section 8.

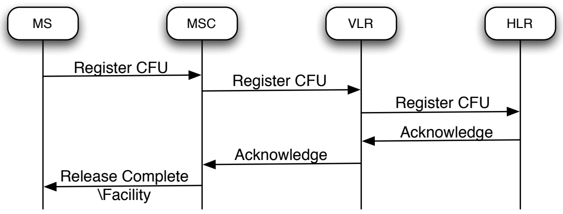

Figure 12: A messaging flow for unconditional call forwarding.

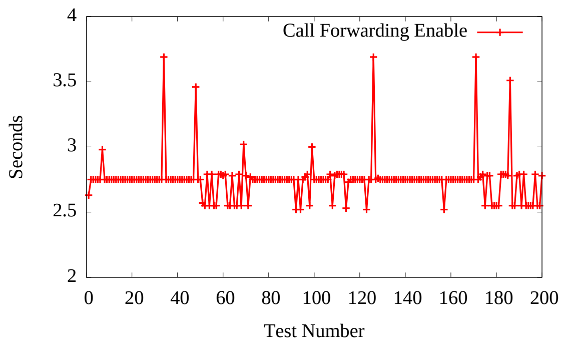

Figure 13: Response times for CF enable requests.

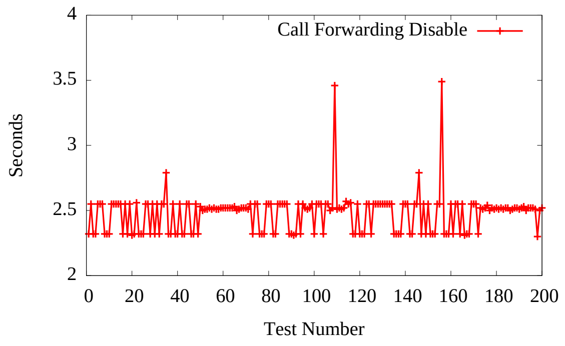

Figure 14: Response times for CF disable requests.

Given an average time of 2.5 seconds and the additional 2 second delay between transmissions, a throughput of 0.22 `update_subscriber_data` requests can be made per second per device. This rate represents a significant improvement over `update_location`.

6.4 Insert/Delete Call Forwarding

This companion set of services allows a user to redirect incoming phone calls to other devices. Such functionality is useful for scenarios such as when a user does not wish to carry their mobile device. The activation and deactivation of this service requires a single exchange with the HLR. As depicted in Figure 12, call forwarding information (e.g., both status and target number) is only stored in the HLR. All requests to both activate and deactivate this service must interact with the HLR directly.

Figures 13 and 14 demonstrate the response times for `insert_call_forwarding` and `delete_call_ forwarding` meta-commands, with respective averages of 2.7 seconds ($\sigma = 0.16$) and 2.5 seconds ($\sigma = 0.15$). While such operations perform very similar functions, the higher response time associated with the activation of call forwarding can be explained via standards documents [1]. Specifically, `insert_call_forwarding` performs additional checks and an extra database read before writing the target number.

These characteristics make `insert_call_forwarding` a good candidate for attack traffic. `delete_call_ forwarding` appears to be an even better candidate due to its lower response time. However, `delete_call_ forwarding` messages can only be sent if call forwarding is already enabled. Attempting to disable call forwarding when it is not enabled results in an immediate error which does not reach the HLR. This results in throughputs of 0.21 and 0.19 requests per second, respectively.

In spite of `update_subscriber_data` having a slightly faster execution time, our experiments show that `insert_call_forwarding` is the most attractive command for an adversary to use. Given the same amount of traffic, the latter meta-command is more expensive for the HLR to process on a per-message basis. *Accordingly, testing on the live network has allowed us to confirm the behavior observed when using TM1 and confirm our selection of* `insert_call_forwarding` *as the most effective request for attacking an HLR.*

7. ATTACK CHARACTERIZATION

Performing tests on a live network allows us to confidently identify the meta-command whose execution creates the most significant processing by the HLR: `insert_call_forwarding`. However, such experiments do not tell us about the attack itself. This section quantifies the impact of such attacks on normal traffic. By identifying the number of requests needed to degrade the HLR's throughput beyond a certain point, we are finally able to realistically estimate the number of infected devices an adversary would require to cause widespread outages.

We model these attacks by modifying TM1's client code. Unlike the single-threaded TM1 clients that attempt to maximize throughput, our multi-threaded malicious clients can coordinate the transmission of specific volumes of traffic. This flexibility allows us to determine the impact of specific attack sizes. We are also able to achieve higher throughput than the standard TM1 clients by aggressively servicing requests without waiting for acknowledgements. The architectural specifics of our malicious clients are discussed in greater detail in the Appendix.

Note that we study the effects of attacks against both normal and very high levels of benign traffic. Almost counter-intuitively, increasing the number of legitimate requests during an attack will often improve the success rate of legitimate clients. Such an increase is the result of an increased probability that legitimate traffic will arrive between malicious requests. This phenomenon is observable in our experiments.

Figure 15 shows the effect of an attack on an HLR running MySQL supporting one million users. As previously mentioned, such systems are representative of many currently deployed HLRs. The impact of attack on legitimate traffic is significant. The lower intensity traffic stream, which achieves a throughput of 2427 default mix TPS, is reduced to a throughput of only 146 default mix TPS at an attack rate of 2500 `insert_call_forwarding` TPS. The high intensity traffic stream, whose throughput is 4132 default mix TPS under normal conditions, is dropped to 273 default mix TPS under a constant attack of 5000 `insert_call_forwarding` TPS. For both cases, the throughput of legitimate traffic is reduced by more than 93% by the presence of attack traffic.

The impact of attacks on a more capable HLR supporting one million users is shown in Figure 16. While the system running SolidDB is certainly capable of maintaining service during small attacks, the performance of these systems can also be extensively degraded. In the high traffic case, throughput is reduced from 5424 to 1340 default mix TPS at an attack rate of 30,000 `insert_call_ forwarding` TPS. The lower traffic scenario is similar impacted

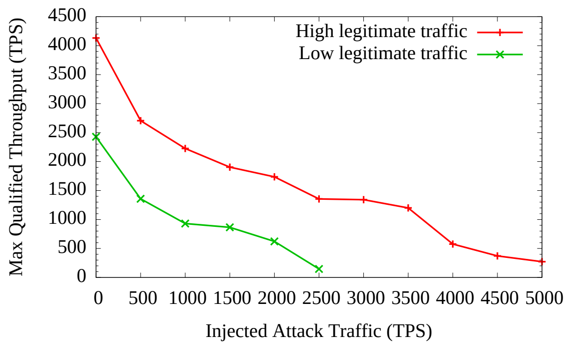

Figure 15: An attack on an HLR running MySQL using TM1's default mix of commands. In both scenarios, throughput of legitimate traffic is reduced by 93%.

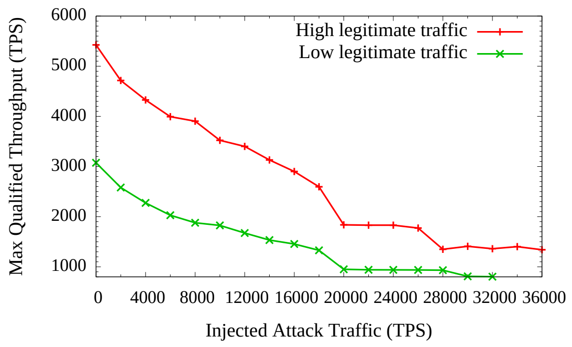

Figure 16: An attack on an HLR running SolidDB using TM1's Default Mix of commands. In both scenarios, throughput of legitimate traffic is reduced by 75%.

and experiences a reduction in throughput from 3075 to 803 default mix TPS at an attack rate of 30,000 `insert_call_forwarding` TPS. These attacks represent a reduction in throughput of approximately 75% in both cases.

From these attacks, it now becomes possible to calculate the number of devices an adversary requires to successfully launch an attack against an HLR. Given the 4.7 second wait between the successful transmission of AT commands in our experiments, an attack on the HLR running the MySQL database under normal conditions would require approximately 11,750 infected mobile phones. Achieving a similar result against a high traffic period would require 23,500 infected phones. Assuming that each of these HLRs service one million users, only 1.2% and 2.4% rates of infection. Attacks on the more capable system would require an increase in the number of compromised devices: 141,000, or an infection rate of 14.1%, for both cases. Because significantly higher infection rates are suspected on desktop machines, such levels of infected devices remain realistic.

Assuming that malcode spreads through some means other than only Bluetooth (e.g., browser-based exploits, executables), infection is unlikely to be limited to devices assigned only to a single HLR. Accordingly, a 14.1% infection rate for a single HLR is likely to reflect a similar rate of compromise for devices served by all HLRs in the network. Given a coordinated attack signal, these networks of compromised phones are more likely to detrimentally affect communications on a nation-wide scale instead of just a single area code. However, as we discuss in the next section, it is still possible for bottlenecks in the wireless portion of the network to prevent such an attack from being carried out.

8. AVOIDING WIRELESS BOTTLENECKS

With a realistic characterization of an attack on the wired portion of a cellular network, a number of obstacles to the successful execution of such an attack exist. In this section, we discuss a number of bottlenecks in the wireless portion of the network that may impede attack traffic. Understanding these choke points provides valuable information about the geographic concentration of compromised nodes. We then examine issues of command and control and consider both the reuse of traditional schemes and the operation of unique cellular techniques.

8.1 Wireless Bottlenecks

The wireless portion of the network is susceptible to congestion due to the limited capacity of two channels - the Random Access Channel (RACH) and the Standalone Dedicated Control Channels (SDCCH). These channels are used as follows: when attempting to perform signaling with the network (i.e., initiate a service), a mobile station will send an access request to the base station over shared-access RACH. If available, the base station assigns an SDCCH, over which the signaling with the HLR itself is performed. We explore the performance characteristics of these channels below.

8.1.1 RACH Capacity

It is possible to experience contention for the RACH as this resource is shared between all devices in an area. We characterize this contention by describing how access is regulated. GSM multiplexes traffic on a single frequency through the use of time division multiple access (TDMA). Channels are superimposed over timeslots, each of which lasts 0.577 ms, in a frame, which contains eight timeslots and lasts 4.615 ms. Devices attempting to iniatite signaling with the network need only gain access to a single RACH timeslot, as all further communications will take place on dedicated channels.

Access to the RACH is governed by the slotted ALOHA protocol, which provides a maximum throughput of:

$$S \quad = \quad Ge^{-G}$$

where throughput is measured as the percentage of successful transmissions, and G is the number of transmission attempts per time slot, also known as the offered load. S is maximized at 37% when $G = 1$.

The offered load, G, also known as ρ, is defined as:

$$\rho \quad = \quad \frac{\lambda}{\mu}$$

where λ is the arrival rate in commands per second and μ^{-1} is the channel hold time. Each request to transmit an attack command can be sent in a single time slot, making $\mu^{-1} = 4.615$ ms. Each service area is therefore capable of supporting:

$$\rho \quad = \quad \frac{1}{0.004615 \text{ s}} * 0.37 \approx 80 \text{ transmissions per second}$$

Knowing that each base station supports three sectors or service areas, each with its own RACH, the attack would need to be distributed over α base stations to deliver such a load to the HLR:

$$\alpha = \frac{5000 \text{ messages/sec}}{3 \text{ sectors/cell} * 80 \text{ RACH transmissions/sec}}$$
$$\alpha = 21 \text{ base stations}$$

Such a small number of towers is reasonable, even when distributed over a relatively small network. For example, if there are $\alpha = 200$ towers per MSC and 10 MSCs per HLR, the network-wide RACH capacity would 481,074 commands per second. Because our attack of 5,000 messages per second is approximately two orders of magnitude smaller than the capacity of such a system, sufficient capacity exists to launch such an attack. If compromised hosts are well distributed across such a network, RACH congestion is unlikely to be the most significant bottleneck. It is still possible that a large number of compromised phones within close proximity may experience competition for such resources. We will analyze the implications of this observation on the spread of malware after analyzing the SDCCH.

8.1.2 SDCCH Limitations

SDCCHs are the primary means of conveying signaling between devices and the network core. Whether for performing a handoff, receiving a text message, setting up a call or authenticating with the network, these channels are critical in the proper operation of these networks. Traynor et al. [59] previously showed that filling SDCCHs with SMS messages effectively blocked signaling in a single cell. Because an elevated signaling load in the core would elicit a larger-scale outage, a successful attack on an HLR is reliant upon not overwhelming this resource.

Sectors in a GSM network typically allocate 8 or 12 SDCCHs. We model the hold time of this channel as 2.7 seconds, based on our experiments on the `insert_call_forwarding` meta-command from Section 6. While this value represents the time required for both RACH and SDCCH access, we feel that it is close enough to allow us to approximate the bottleneck in this portion of the network. We calculate the number of base stations over which such an attack must be distributed as follows:

$$\rho_{SDCCH} = \frac{1}{2.7} = 0.37$$
$$\alpha = \frac{msgs/sec}{sectors * SDCCHs * \rho_{SDCCH}}$$
$$\alpha = \frac{5,000}{3 * 12 * 0.37} = 375 \text{ base stations}$$

If a service provider deploys more capable HLRs, which our simulations show can handle attack loads of up to 30,000 transactions per second, 2,252 base stations will be needed to handle the air interface load of an attack. Given that providers such as Verizon Wireless and AT&T Wireless are estimated to own or lease upwards of 30,000 towers each [57], even with SDCCH limitations the distribution of phones over base stations needed for a successful attack is feasible.

From these calculations, the characteristics of the wireless bottlenecks become more obvious. While both the RACH and SDCCHs can act as bottlenecks, SDCCHs are a far more limiting resource. If compromised nodes are evenly distributed throughout the network, such contention is unlikely to play a significant role in reducing the amount of attack traffic that reaches the HLR. However, malware that spreads through means of proximity (e.g., Bluetooth-worms [19]) is less likely to be a successful propagator of such an attack. Without significant dispersion of compromised devices, *infections that are highly geographically localized are more likely to accidentally cancel themselves out rather than impact the network on a large scale.* Coordination of thousands of dispersed compromised phones and avoiding contention when high concentrations of such devices are present must therefore be considered.

8.2 Command and Control

A botmaster must develop a means of communicating with and coordinating the actions of these compromised hosts to avoid the above bottlenecks. Improving and disrupting command and control structures and mechanisms on the Internet are active areas of research [11, 21, 20]; however, cellular networks provide significant new challenges and opportunities for command and control. We provide a brief overview of a number of approaches that could potentially be used to manage cellular botnets.

Internet Coordination: Given that an increasing number of mobile phones can connect to the Internet, an adversary could potentially reapply many of the command and control techniques currently being used in that domain to cellular botnets. For instance, devices could easily poll a known communication channel (e.g., IRC, webpage). Adversaries attempting to avoid monitoring and blocking may instead attempt peer-to-peer communication. To reduce control overhead, such methods may be used in concert with time triggered attacks. However, all of these standard techniques may potentially be constrained by the architecture of the network itself. Because these networks are not designed to handle a large number of concurrent flows [60], nodes serviced by the same tower may suffer from low throughput and high delays when attempting to access the communication channel. Additionally, because all traffic from the cellular botnet can be observed by the provider, even P2P structures may be easily identified and blocked [36].

Local Wireless Coordination: The highly capable handsets that enable phone-originated attacks also enable advanced command and control methods. Nearly all new mobile devices possess Bluetooth radios, through which direct communications between multiple compromised phones can occur. The advantage to such an approach is that many of the issues facing the use of traditional command and control approaches (e.g., monitoring and network bottlenecks) can be avoided. Such communications, however, are significantly limited by range. Worse still, increasing the density of infected devices to improve connectivity will increase contention for local resources and may limit an attack to a single sector.

The increasing number of mobile devices equipped with 802.11 wireless radios greatly reduces the range issue. Moreover, because such devices can associate with traditional wireless access points, an adversary with an established botnet command and control infrastructure can not only manage compromised cellular devices as part of their larger network, but also use this connection from the Internet to attack the core of the cellular network. Because both of these means of communication are separate from cellular networks, a botmaster would need to consider coverage issues before relying on either method.

Indirect Local Coordination: As previously mentioned, high rates of infection within individual cells may cause significant contention for resources and decrease the effective attack throughput. Figure 17 shows such contention for channels such as the RACH[5]. Even without directly communicating with the other nearby infected devices, it is possible to coordinate and improve the throughput of attack traffic.

The GSM MAC layer includes a back off algorithm that is triggered when a network request times out, but its behavior is tuned for normal traffic patterns, rather than the sudden ramp up and sustained level of traffic characteristic of an attack. The GSM back off algorithm sets a phone's back off timer to a randomly chosen inte-

[5]The details of the simulator used to model these attacks are included in the Appendix.

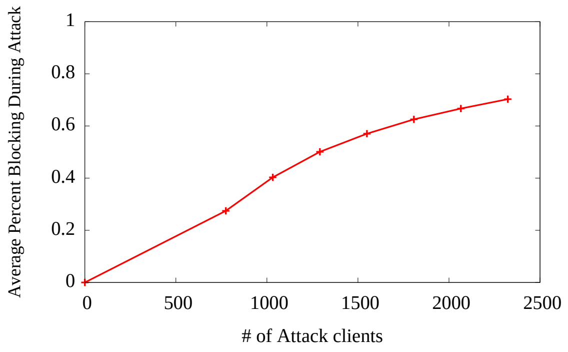

Figure 17: Percentage blocking for all messages on the RACH as attack traffic increases.

ger multiple of 10 ms, where the minimum and maximum values for the back off timer are set as network parameters. The traffic patterns generated by an attack require a more reactive back off algorithm:

$$\begin{aligned}\beta_{avg} &= \alpha\beta_{inst} + (1-\alpha)\beta_{avg}\\ \textbf{if } \beta_{inst} &> \beta_{avg} \textbf{ then}\\ \gamma &= 2\gamma\\ \textbf{else if } \beta_{inst} &< \beta_{avg} \textbf{ then}\\ \gamma &= \gamma/2\end{aligned}$$

Where β_{inst} is the instantaneous measurement of round trip time, and β_{avg} is the running average. α is a weight parameter set to 0.5, and γ is the back off timer. This algorithm uses multiplicative increase and decrease to rapidly react to changing channel conditions. The triggering of an attack leads to the sudden onset of a large amount of traffic, requiring a very aggressive back off algorithm. Further development and simulation of the exponential back off algorithm is outside the scope of this paper and has been left as future work.

Due to its use of local feedback, exponential back off reduces congestion in areas with a high density of compromised phones more effectively than time or network triggering. Unfortunately, reducing RACH contention using exponential back off leads to unavoidable inefficiencies in the use of available network resources.

9. ATTACK MITIGATION

Given the vastly expanded programmability of these systems, coupled with their increased connectivity to external data networks, new defenses must be implemented.

Many cellular networks perform database replication in order to avoid widespread outages if HLR hardware fails. Should such a situation occur, traffic is rerouted to another HLR with access to the backup copy of the database. The impact of attacks against single HLRs is lessened assuming that the other HLR can support the additional load [48]. However, such a defense is unlikely to provide additional protection against a large-scale attack. In particular, an attack targeting a large portion of the HLRs in a single network is unlikely to be mitigated by migrating attack traffic to other servers. Instead, the failure of any one HLR may lead to a significant increase of traffic to one or more of the others. As a number of providers consider an architecture in which a small number of centrally located, highly capable HLRs process traffic for the entire network, such attacks may in fact be more likely to succeed.

Filtering can to help protect against some variants of this attack. Because `insert_call_forwarding` is not critical to the basic functioning of the network, such filters could be aggressively tuned without much worry of the impact of false positives. *Call gapping*, a technique commonly used for call overload situations for expected events, could potentially be adapted for signaling overload attacks. For such a defense to be effective, a means of rapidly deploying shedding rules corresponding to a specific attack and not simply benign elevated traffic conditions would need to be further investigated.

Basic filtering and shedding are two examples of possible network defenses a service provider can implement. However, developing mechanisms intelligent enough to respond to a more dynamic attack remains challenging, especially if adversaries deliberately attempt to target nodes other than the HLR. In particular, because a significant amount of context is lost as messages move between mobile devices and the HLR (e.g., granularity of location), it will likely become difficult for a provider to separate attacks from other traffic. Moreover, because of the large overhead associated with the first hop of communications in such networks, filtering in the core may occur too late to prevent users from experiencing significant congestion. Preventing such devices from ever transmitting malicious traffic is arguably more critical in this environment than the traditional Internet setting. Accordingly, much work remains to be done in the development of defenses.

10. CONCLUSION

By providing users with highly limited mobile devices and restricting connectivity with external systems, cellular networks have long been able to appreciably limit the potential for malicious behavior. However, the arrival of highly capable mobile phones, most of which lack basic security mechanism, significantly diminishes such protections. In this paper, we have demonstrated that even relatively small botnets composed entirely of mobile phones pose significant threats to the availability of these network. Depending on the specific hardware deployed by a provider, our results demonstrate the potential to cause nation-wide outages with only single-digit infection rates. Moreover, because such attacks work by quietly launching network service requests and not through a flood of phone calls, users are unlikely to be aware of their execution. While command and control is potentially more challenging in this environment, such obstacles can be overcome through the use of a combination of the multiple network interface features common to most mobile devices.

While similar attacks have been studied in the context of the Internet, the work in this paper should be viewed as a warning of the increasingly sophisticated attacks possible in telecommunications infrastructure. By demonstrating that core elements of the network are now addressable, this work identifies the potential for attacks attempting to compromise specific components. Such threats must be considered proactively in order to maintain the reliability long associated with these networks.

Acknowledgments

We would like to thank Chaitrali Amrutkar, William Enck, Mike Hunter, Fabian Monrose and Frank Park for their invaluable comments. This work was supported in part by the NS National Science Foundation (CNS-0721579, CNS-0905447, CNS-0721579 and CNS-0916047). Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

11. REFERENCES

[1] 3rd Generation Partnership Project. Call forwarding (CF) supplementary services. Technical Report 3GPPTS23.082 v7.2.0.

[2] 3rd Generation Partnership Project. General packet radio service (GPRS). Technical Report 3GPP TS 23.060 v8.0.0.

[3] 3rd Generation Partnership Project. Mobile Application Part (MAP) Specification. Technical Report 3GPP TS 29.002 v8.7.0.

[4] 3rd Generation Partnership Project. Physical layer on the radio path; General description. Technical Report 3GPP TS 04.18 v8.26.0.

[5] Altibase Performance Solutions. Home Location Register Management System. http://www.altibase.com.cn/down_load/engpdf/app_communication.pdf, 2002.

[6] J. Barta, J. Moody, and C. Cranton. Nortel Networks Selected as Home Location Register Provider to Cingular Wireless. http://www.nortel.com/corporate/news/newsreleases/2003b/06_25_03_cingular_wireless.html, 2003.

[7] S. Berinato. Online Extortion – How a Bookmaker and a Whiz Kid Took On an Extortionist and Won. *CSO Online*, May 2005.

[8] V. J. Bono. 7007 Explanation and Apology. http://www.irbs.net/internet/nanog/9704/0440.html, 1997.

[9] A. Bose and K. Shin. On mobile viruses exploiting messaging and bluetooth services. In *Proceedings of the IEEE International Conference on Security and Privacy in Communication Networks (SecureComm)*, 2006.

[10] S. Byers, A. Rubin, and D. Kormann. Defending Against an Internet-based Attack on the Physical World. *ACM Transactions on Internet Technology (TOIT)*, 4(3):239–254, August 2004.

[11] E. Cooke, F. Jahanian, and D. McPherson. The Zombie roundup: Understanding, detecting, and disrupting botnets. In *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet (SRUTI) Workshop*, 2005.

[12] D. Dagon, M. Antonakakis, K. Day, X. Luo, C. P. Lee, and W. Lee. Recursive DNS Architectures and Vulnerability Implications. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2009.

[13] N. A. El-Fishway, M. A. Nofal, and M. Tadros. An Improvement on Secure Communication in PCS. In *Proceedings of the IEEE Performance, Computing, and Communications Conference*, 2003.

[14] F-Secure Corporation. F-Secure Computer Virus Descriptions: Cabir. http://www.f-secure.com/v-descs/cabir.shtml, December 2004.

[15] F-Secure Corporation. F-Secure Computer Virus Descriptions : Mabir.A. http://www.f-secure.com/v-descs/mabir.shtml, April 2005.

[16] F-Secure Corporation. F-Secure Computer Virus Descriptions : Skulls.A. http://www.f-secure.com/v-descs/skulls.shtml, January 2005.

[17] F-Secure Corporation. F-Secure Malware Information Pages: Worm:SymbOS/Commwarrior. http://www.f-secure.com/v-descs/commwarrior.shtml, 2008.

[18] R. Farrow. DNS Root Servers: Protecting the Internet. *Network Magazine*, 2003.

[19] C. Fleizach, M. Liljenstam, P. Johansson, G. M. Voelker, and A. Mhes. Can You Infect Me Now? Malware Propagation in Mobile Phone Networks. In *Proceedings of the ACM Workshop on Recurring Malcode*, 2007.

[20] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection. In *Proceedings of the USENIX Security Symposium (SECURITY)*, 2008.

[21] G. Gu, J. Zhang, and W. Lee. BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2008.

[22] C. Guo, H. J. Wang, and W. Zhu. Smart Phone Attacks and Defenses. In *Proceedings of Third ACM Workshop on Hot Topics in Networks (HotNets-III)*, 2004.

[23] N. Gupta. Enabling High Performance HLR Solutions. http://www.sun.com/products-n-solutions/hw/networking/atca/HLR-on-ATCA-v2-Final.pdf, 2006.

[24] C. Haney. NAI is latest DoS victim. http://security.itworld.com/4339/NWW116617_02-05-2001/page_1.html, February 5 2001.

[25] J. Hedden. 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.

[26] P. Howard, M. Walker, and T. Wright. Towards a coherent approach to third generation system security. In *Second International Conference, 3G Mobile Communication Technologies*, 2001.

[27] D. Ilett. Symantec website under DDoS attack. http://software.silicon.com/malware/0,3800003100,39150478,00.htm, 2005.

[28] J. Ioannidis and S. Bellovin. Implementing Pushback: Router-Based Defense Against DDoS Attacks. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, February 2002.

[29] A. Juels and J. G. Brainard. Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 1999.

[30] A. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *Proceedings of ACM SIGCOMM*, 2002.

[31] Kingping 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.

[32] K. Kotapati, P. Liu, and T. La Porta. CAT A Practical Graph & SDL Based Toolkit for Vulnerability Assessment of 3G Networks. In *Proceedings of the IFIP TC-11 International Information Security Conference, Security and Privacy in Dynamic Environments*, 2006.

[33] M. Lad, R. Oliveira, B. Zhang, and L. Zhang. Understanding Resiliency of Internet Topology Against Prex Hijack Attacks. In *Proceedings of the IEEE/IFIP Conference on Dependable Systems and Networks (DSN)*, 2007.

[34] P. Lee and T. Woo. On the Detection of Signaling DoS Attacks on 3G Wireless Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 2007.

[35] C.-C. Lo and Y.-J. Chen. Secure communication mechanisms for GSM networks. *IEEE Transactions on Consumer Electronics*, 45(4), 1999.

[36] P. McDaniel, S. Sen, O. Spatscheck, J. V. der Merwe, B. Aiello, and C. Kalmanek. Enterprise Security: A Community of Interest Based Approach. In *Proceedings of the Network & Distributed System Security Symposium (NDSS)*, 2006.

[37] J. Mirkovic and P. Reiher. A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[38] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the Slammer Worm. *IEEE Security and Privacy*, 1(4), July 2003.

[39] C. Mulliner and G. Vigna. Vulnerability Analysis of MMS User Agents. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2006.

[40] C. Pettey. Gartner Says Worldwide Mobile Gaming Revenue to Grow 50 Percent in 2007. http://www.gartner.com/it/page.jsp?id=507467, 2007.

[41] A. C. Popescu, B. J. Premore, and T. Underwood. Anatomy of a Leak: AS9121. http://www.nanog.org/mtg-0505/underwood.html, 2004.

[42] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All Your iFRAMEs Point to Us. In *Proceedings of the USENIX Security Symposium (SECURITY)*, 2008.

[43] R. Racic, D. Ma, H. Chen, and X. Liu. Exploiting Opportunistic Scheduling in Cellular Data Networks. In *Proceedings of the Networking and Distributed Systems Security Symposium*, 2008.

[44] M. Richtel. Yahoo Attributes a Lengthy Service Failure to an Attack. *The New York Times*, February 8 2000.

[45] RIPE Network Coordination Centre. RIPE NCC DNS Monitoring Services. http://dnsmon.ripe.net/dns-servmon/domain/plot?domain=root&day=5&month=2&year=2007&hour=16&period=48h&plot%2F=SHOW, 2007.

[46] RIPE Network Coordination Centre. YouTube Hijacking: A RIPE NCC RIS case study. http://www.ripe.net/news/study-youtube-hijacking.html, 2008.

[47] P. Roberts. Al-Jazeera Sites Hit With Denial-of-Service Attacks. *PCWorld Magazine*, March 26 2003.

[48] P. Roberts. Cingular GSM network downed. http://www.infoworld.com/article/03/09/29/HNcingulardowned_1.html, 2003.

[49] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *Proceedings of ACM SIGCOMM*, pages 295–306, October 2000.

[50] J. Serror, H. Zang, and J. C. Bolot. Impact of paging channel overloads or attacks on a cellular network. In *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, 2006.

[51] Smart Computing Encyclopedia. Smart Computing Encyclopedia - Hayes. http://www.smartcomputing.com/editorial/dictionary/detail.asp?guid=&searchtype=&DicID=17562&RefType=Encyclopedia, 2008.

[52] Solid Information Technology. *Telecom One (TM1) Benchmark Test Suite Guide Version 3.0-1*, November 2006.

[53] Solid Information Technology. Frequently Asked Questions about the TM1 Benchmark. http://www.solidtech.co.jp/developers/CarrierGrade/TM1/TM1_FAQ.pdf, 2008.

[54] Solid Information Technology. The TM1 Benchmark Results. http://www.soliddb.com/TM1Results, 2008.

[55] A. Stavrou, D. L. Cook, W. G. Morein, A. D. Keromytis, V. Misra, and D. Rubenstein. WebSOS: An Overlay-based System For Protecting Web Servers From Denial of Service Attacks. *Journal of Computer Networks*,

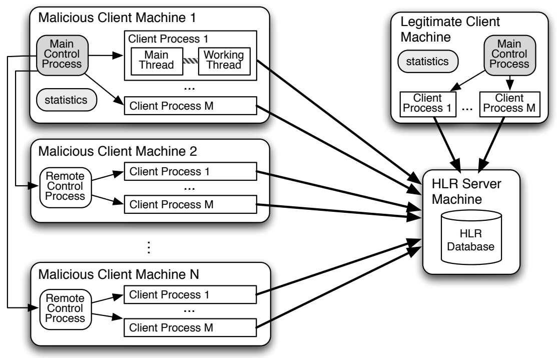

Figure 18: Modified TM1 architecture

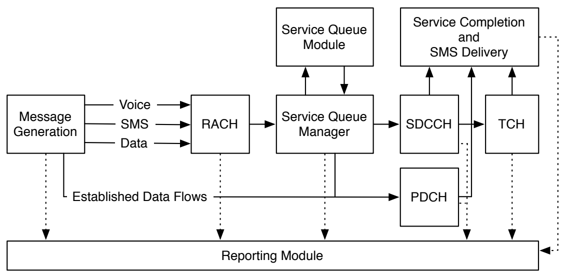

Figure 19: Simulator architecture

48(5), 2005.

[56] A. Stavrou, A. Keromytis, J. Nieh, V. Misra, and D. Rubenstein. MOVE: An End-to-End Solution To Network Denial of Service. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2005.

[57] Steel In The Air, Inc. Total Cell Towers and Cell Sites by Provider. `http://www.steelintheair.com/Blog/2005/08/total-cell-towers-and-cell-sites-by.html`, 2005.

[58] P. Traynor, W. Enck, P. McDaniel, and T. La Porta. Mitigating Attacks on Open Functionality in SMS-Capable Cellular Networks. *IEEE/ACM Transactions on Networking (TON)*, 2009.

[59] P. Traynor, W. Enck, P. McDaniel, and T. F. La Porta. Exploiting Open Functionality in SMS-Capable Cellular Networks. *Journal of Computer Security (JCS)*, 2008.

[60] P. Traynor, P. McDaniel, and T. La Porta. On Attack Causality on Internet-Connected Cellular Networks. In *Proceedings of the USENIX Security Symposium*, 2007.

[61] Trifinite. BlueBug. `http://trifinite.org/trifinite_stuff_bluebug.html`, 2004.

[62] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Proceedings of Eurocrypt*, pages 294–311, 2003.

[63] L. von Ahn, M. Blum, and J. Langford. Telling humans and computers apart automatically. *Communications of the ACM*, 47(2):56–60, 2004.

[64] J. Wang, X. Liu, and A. A. Chien. Empirical Study of Tolerating Denial-of-Service Attacks with a Proxy Network. In *Proceedings of the USENIX Security Symposium*, 2005.

Appendix

TM1 Client Modifications

We modified TM1 to enable control over the rate at which our malicious clients inject attack traffic into the HLR. Our modification involves adding a new test parameter, request rate, to the configuration file, which will be parsed by the main control node, and passed to the remote control nodes and finally to all the client processes. We also altered major part of the client process, which contains approximately 4400 lines of code, to allow multi-threading.

To regulate the attack traffic, the desired amount of attack traffic is divided by the number of available attack client machines. The remote control node at each client machine in turn splits the assigned amount of attack traffic among the client processes running on the machine. In each process, the main thread spawns working thread(s) and creates the transaction request at the assigned request rate. It randomly selects the transaction type based on the percentage defined in the configuration file and puts such request into the queue. One or more working threads continuously retrieve requests from the queue at a time (based on the availability of the requests in the queue), prepare the SQL request, submit it to the HLR, and wait for the result. They are also responsible for transaction rollback and commit.

Each working thread creates a connection to the HLR when it is initialized. This connection is maintained throughout its lifetime (as with the original TM1, which each client maintains the database connection throughout the run). Each thread also maintains a copy of the transaction data structure and global variables table to avoid contention and deadlock.

GSM Simulator Architecture

We use the GSM simulator built in [60] to demonstrate RACH throughput. 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 19, 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 [25]. The path taken by individual requests depends on the flow type. We focus on the data path as the behavior of SMS and voice messages were explained in the previous iteration of the simulator.

If the network has not currently dedicated resources to a flow on the arrival of a packet, it is passed to the RACH module. This random access channel is implemented in strict accordance with 3GPP TS 04.18 [4] and is tunable via `max_retrans` and `tx_integer` values.

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.