On Mobile Viruses Exploiting Messaging and Bluetooth Services

Abhijit Bose and Kang G. Shin
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122
Email: {abose, kgshin}@eecs.umich.edu

Abstract— **The exponential growth of mobile messaging worldwide has made it an indispensable tool for social and business interactions. The interoperability between SMS (Short Messaging Service) and IM (Instant Messaging) networks has enabled mobile users to communicate over the Internet seamlessly. However, the proliferation of cellular phones and handheld devices with messaging capability has also attracted virus writers who increasingly develop malware targeted to mobile handheld devices. The mobile viruses discovered so far have exploited vulnerabilities in Bluetooth to infect a nearby device and then use SMS to spread itself to other devices in the mobile network. This problem is expected to become worse with the growth of MMS (Multimedia Messaging Service), mobile games, mobile commerce and peer-to-peer file-sharing in near future. We investigate the propagation of mobile worms and viruses that spread primarily via SMS/MMS messages and short-range radio interfaces such as Bluetooth. First, we study these vulnerabilities in-depth so that appropriate malware behavior models can be developed. Next, we study the propagation of a mobile virus similar to Commwarrior in a cellular network using data from a real-life SMS customer network. Each handheld device is modeled as an autonomous mobile agent capable of sending SMS messages to others (via an SMS center), and is capable of discovering other devices equipped with Bluetooth. Since mobile malware targets specific mobile OSs, we consider diversity of deployed software stacks in the network. Our results reveal that hybrid worms that use SMS/MMS and proximity scanning (via Bluetooth) can spread rapidly within a cellular network, making them potential threats in public meeting places such as sports stadiums, train stations, and airports.**

I. Introduction

With the growing popularity of mobile devices such as smart phones, handsets and PDAs, the mobile Internet is now a key component of many enterprise and social networks. It is also quickly becoming a major channel for distributing digital media content such as music, video and advertising, and for enabling m-commerce activities. With the proliferation of mobile devices, there is, however, an increasing threat from mobile malware (i.e., viruses, worms, spams and other malicious software), that targets these devices, using traditional social-engineering techniques such as email and file-sharing, as well as vectors unique to mobile devices such as Bluetooth and SMS (Short Messaging Service) messages. Mobile viruses targeting cellular phones, PDAs and Bluetooth-enabled devices have already started to appear [1], [2]. Studying such viruses — their capabilities, infection models and vulnerabilities they typically exploit — is therefore an important area of research.

The mobile viruses discovered so far have caused little damage as they require explicit user interaction for installation and activation. However, potential harm from future malicious agents can be severer in the form of handset downtime, service disruption due to Denial-of-Service (DoS) attacks, physical damage to device hardware, and theft of sensitive data on the device. Similar to email viruses, these agents may also target SMS/MMS services for distributing spam and phishing messages. There are several factors that make mobile devices particularly vulnerable to future mobile viruses. First, recognizing customer demand for data-rich cellular services, carriers around the world have been deploying 3G (third generation) cellular systems at a rapid pace. Currently, there are more than 130 3G networks [3] (WCDMA and CDMA2000 1X EV-DO) worldwide. Many of these networks offer real-world data rates of 1.4 Mbps and 128 Kbps for download and upload, respectively. The download data rates are expected to rise to 7.3 Mbps in early 2008 and 10.2 Mbps in 2009. At these rates, mobile users will be able to run many feature-rich applications on their mobile devices that traditionally require access to a high-speed enterprise network. The processing power (CPU speed and storage capacity) of handheld devices is also increasing rapidly. Many smart phones [4] already contain a full-fledged OS like Symbian, Windows Mobile and Palm OS, allowing users to download a wide variety of applications. Almost all of these OSs support services such as email, SMS/MMS, and application development in C++ and Java. Consequently, the malware writers increasingly find it easier to generate device-generic but vulnerability-specific malware for mobile devices. As a result, the current count of known mobile malware stands at 100, up from only 10 in previous years combined.

The majority of the published studies on modeling and containment of malware have focused on scanning- and email-worms due to their prevalence and several successful large-scale attacks on the Internet. On the contrary, there appears to be very little published work on mobile viruses and worms. This is the primary motivation of our work. We study the mobile viruses discovered to date and the various target discovery, infection and replication mechanisms they deploy to spread to other devices. Since the majority of these viruses have so far exploited Bluetooth and SMS/MMS services on cellular handsets, we investigate vulnerabilities in these two vectors

A. Bose is currently with IBM T.J. Watson Research, Hawthorne, NY. This work was supported in part by NSF under Grant No. CNS 0523932.

in-depth. We then model the propagation of a mobile virus similar to Commwarrior that targets Bluetooth and SMS/MMS services, using a fine-grained agent-based malware simulator that we develop for studying mobile worms. Using data from a real-world SMS network, we emulate the propagation of this virus in a small mobile network representative of a public meeting place such as a stadium or airport.

This paper makes two primary contributions. First, it provides a detailed analysis of vulnerability from mobile viruses. Second, it proposes a new emulation framework for studying propagation models and containment strategies of viruses in mobile environments. The paper is organized as follows. In Section II, we provide an overview of mobile viruses and vulnerabilities. The specific vulnerabilities associated with Bluetooth and SMS/MMS messaging services are discussed in Sections III and IV, respectively. Next, we discuss briefly our agent-based simulation framework in Section V. Section VI presents the results of our simulation studies using data from a real-world SMS network. Finally, we make concluding remarks in Section VII.

II. Analysis of Mobile Viruses

Malicious agents that specifically target mobile phones and handheld devices are on the rise. The earliest versions of these were considered harmless since these were not written to spread from one device to another. The most recent mobile malware, however, are capable of spreading to nearby devices via Bluetooth, and, therefore, pose a more serious threat to enterprise networks. In what follows, we list the most common spreading mechanisms, target platforms, and client vulnerabilities of mobile malware discovered to date. Wherever possible, we classify them as *virus*, *worm* or *trojan* to indicate their replication and propagation mechanisms. Further details on them are available on a number of security-vendor web sites.

- One of the earliest viruses written for handheld devices (Palm PDAs), the PalmOS.Liberty.A virus (2001), had to be manually installed and executed for it to become active. The virus deleted all applications and databases on a Palm OS-compatible device. The Liberty virus and other similar trojans by their design are not likely to spread quickly due to their manual infection process and therefore, represent a relatively low threat.
- A virus for Palm OS, called Phage (2000), was conceived mostly as a demonstration — it could spread from one PDA to another if infected files were shared via infrared beaming or a docking station. This was an improvement from manual infection.
- The Spanish Timophonica [5] worm (2000) was programmed to send SMS messages to random GSM phone numbers via a specific SMS gateway. Only reported in Spain, this worm represented the beginning of more advanced mobile viruses to come, since it modified MS Outlook settings and the device registry of an infected phone. If the accompanied trojan code was successfully installed, it also deleted CMOS memory and Master Boot Records of the device. It could propagate via the Outlook email client using stored address book entries. Worms such as Timophonica are early examples of hybrid mobile malware since they can spread via both wireless and wired networks.
- The Japanese 110 worm (2000) took advantage of a vulnerability in the NTT DoCoMo i-mode mobile phones. This phone has a capability similar to "mailto:" available in html — users can automatically dial a number by clicking on the linked number contained in an email or web page. Therefore, individual phone numbers in the address book can become victims of DoS attacks.

More recent mobile agents have targeted Nokia series 60 cell phones running Symbian OS due primarily to their popularity and advanced features, such as Bluetooth and SMS/MMS services. These agents can search for nearby Bluetooth-enabled devices using *proximity scanning*. Note that proximity scanning requires physical proximity (e.g., up to 10 meters for Bluetooth Class-2 devices) between an infected device and a target device, whereas SMS or MMS requires only a network connection between an infected device and the service gateway for sending messages and malicious payload to other devices. Similar to email viruses, the mobile viruses use social-engineering techniques to entice unsuspecting users to click on infected audio, video or picture attachments. Some examples are:

- The Mabir (2004) worm spreads by selecting addresses of newly-received MMS messages. The primary damage from Mabir is the transmission of MMS messages.
- Cabir (2004–2005) and its variants replicate over Bluetooth connections, and install the worm payload as a Symbian System Installation (SSI) file. The Cabir worm drains the power of the infected phone as it continually scans for other Bluetooth devices nearby. An outbreak of Cabir was reported at the 2005 world athletics championships in Helsinki, Finland, affecting Nokia cell phones.
- Lasco (2005) propagates by transferring its payload to any device in range. It attaches itself to SSI files on the compromised device. Lasco combines the self-replication of viruses with the self-propagation capability of worms.
- Commwarrior (2005) is another worm that propagates by sending messages (along with the payload as attachments) to an MMS-enabled phone number randomly chosen from the compromised device's address book, and resets the infected device on the first hour of 14-th of any month. Once it infects a phone, it starts searching for nearby Bluetooth devices for sending infected files. We will use Commwarrior as a typical mobile worm in our simulation studies presented in Section VI. A similar malware, the Minuka trojan (2004), finds targets from SMS address books at predetermined web sites. Mos (2005) is a variant of Minuka that dials a high-cost phone number (e.g., 1-900).
- Skulls (2005) is a trojan that propagates by sending both SMS and MMS messages, and overwrites many default phone applications such as the address book, and e-mail viewer and to-do lists. Many variants of Skulls have been observed in the wild.
- Drever (2005) is a trojan that propagates by prompting a user to install an update for Symbian OS. The primary

damage from this trojan is disabling Symbian antivirus programs (SimWorks) on the device.
- Locknut (2005) is another trojan that propagates similar to Lasco, but overwrites ROM binaries and may crash the OS. It can also drop variants of Cabir on the infected device.
- *Cardblock* (2005) is the first known malware (of type trojan) to attack MultiMedia Cards (MMC) flash memory of mobile phones — it is a trojanized version of Symbian application InstantSis that allows users to repack already-installed SIS files and copy them to another phone. However, when users try the trojanized version, a payload blocks the memory card by setting a random password to it and deletes critical system and mail directories.
- The *Redbrowser* trojan (2006) is the first malware targeting J2ME (Java 2 Mobile Edition) phones and represents a major evolution in mobile viruses. Instead of focusing on high-end smart phones running on Symbian or Pocket PC, it works on many low-end phones with J2ME support. Redbrowser pretends to be a WAP browser offering free WAP browsing and SMS messages — the purpose is to use a social engineering technique to fool the user into sending SMS messages. However, it actually sends a flood of SMS messages to a specific number and therefore, can cause financial damage to the user.

The example of Redbrowser shows that the emergence of mobile viruses is not unique to Symbian OS. Another malware, WinCE.Duts (2005) is the first proof-of-concept Windows CE (Pocket PC) virus infecting only ARM-based devices. It simply appends itself to executable files (*.EXE) in the root folder of the device and modifies the program execution header so that the payload runs first. The virus takes advantage of a vulnerability in the "coredll" API of Windows CE for appending itself to executables.

A. Crossover Malware

Before we discuss potential vulnerabilities of Bluetooth and SMS/MMS services that can be exploited for malicious purposes, we must mention an emerging class of malware called "crossover" infectors that can spread from mobile devices to desktop PCs, or vice versa. For example, Cardtrap.A (2005) is a Symbian SIS file trojan that disables a number of applications on Nokia series 60 cell phones in addition to installing variants of skulls and Cabir. However, the most significant characteristics of Cardtrap is that it also installs three Windows worms (Win32.Rays, Win32.Padobot.Z and Win32.Cydog.B) onto the device's memory card. Once the card is inserted into the PC, Padobot.Z will attempt to start automatically on machines running Windows OS via the "autorun.ini" file. However, since Windows does not generally support autorun from a memory card, the current version depends on user interaction to be launched.

Conversely, a recent virus called Crossover (2006) spreads from Windows desktop PCs to mobile devices running on Windows Mobile Pocket PC. Once it is installed on a Windows PC, the virus makes a copy of itself and adds a registry entry pointing to the new file so that the payload is activated each time the machine is rebooted. It then waits for an application (e.g., ActiveSync) for synchronizing Pocket PC devices with the infected Windows desktop PC. When a connection is detected, it copies itself over to the Pocket PC device, deletes all files in the *My Documents* directory, copies itself to the system directory and places a link to itself in the startup directory. Current-generation crossover viruses such as Cardtrap and Crossover are not powerful enough to cause concern at the present. Nonetheless, these viruses give rise to the future possibility of an attacker using a Bluetooth- or SMS/MMS-capable handset to transfer a malicious agent designed to spread to millions of desktop PCs on the Internet.

III. Bluetooth Vulnerabilities

Bluetooth is a short-range radio technology, providing wireless connectivity in ranges from 10 m (Class 2) to 100 m (Class 1), with throughput up to 723.2 Kbps, or 2.1 Mbps (with enhanced data rates introduced in 2005) using the globally available 2.4 GHz radio band. Each device can simultaneously communicate with up to 7 other devices to form a piconet. Since the Bluetooth Special Interest Group (SIG) [6] was founded in 1998, a number of enhancements, such as synchronous modes for voice, piconet formation and enhanced data rates, have been made to the original standard, making it a popular choice for interconnecting mobile and handheld devices. One industry research report estimates that nearly 300 million devices were shipped with Bluetooth enabled in 2005. A recent study from IDC estimates that there will be over 922 million Bluetooth-enabled devices worldwide by 2008. The primary application area of Bluetooth to date has been in mobile and wireless consumer products such as mobile phones (nearly 60% of the market), headsets, mobile printing, hands-free control (e.g., in cars), computer peripherals (e.g., keyboard and mouse), and for synchronization of handheld devices. With the recent adoption of ultra wideband (UWB) radio [7] by the Bluetooth SIG, Bluetooth devices will soon be able to transmit data at a rate similar to USB and firewire, and will allow transmission of high-definition video to other mobile devices and files for digital music players.

A number of potential vulnerabilities and exploits have been identified with Bluetooth devices. The authors of [8] proposed an analytical model called probabilistic queuing for modeling malware spreading in an ad-hoc Bluetooth environment. Although their model does not consider fading effects and Bluetooth software stack issues, such a model can be useful to estimate the final size of an epidemic involving a mobile Bluetooth virus. The Common Vulnerabilities and Exploits (CVE) [9] and National Vulnerability Database [10] have recorded a sharp increase in reported vulnerabilities for several popular Bluetooth software stacks. The majority of the resulting exploits are due to programming flaws and incorrect implementation of Bluetooth protocols for pairing, data transfer and device discovery. We first provide a brief description of the Bluetooth link-layer security model, and then review the major vulnerabilities.

The simplest security mechanism provided by Bluetooth is a *device discovery* mode in which the device is either "discoverable" (i.e., visible to other nearby devices) or "non-discoverable." When a mobile virus or worm uses proximity

scanning to search for nearby devices, a "non-discoverable" device has *theoretical* protection against a Bluetooth-based attack. It is theoretical because there are techniques to search for devices that are in non-discoverable mode, as we will describe shortly. Even a non-discoverable device is still visible to previously-paired devices and users who are familiar with its Bluetooth MAC address. The device discovery mode with current-generation Bluetooth devices has several implementation issues. The majority of the mobile phones are enabled with their Bluetooth interface always discoverable by default. Some Bluetooth headsets never return the interface to a non-discoverable mode after closing a connection with a paired phone. The need to be discoverable for pairing with another device can also be exploited by an attacker to learn of nearby devices. Therefore, setting the discovery mode of a Bluetooth device to non-discoverable does not guarantee strong protection against mobile malware.

The basic security mechanism provided by Bluetooth is *link-level authentication*. A link is defined as a communication channel established between two Bluetooth devices. In the link establishment protocol, a *link key* is used to verify the authenticity of the two devices. The link keys are also used to generate *link encryption keys* that control the encryption of data sent over the established link. The procedure by which the two devices establish a shared secret is also known as *pairing*. The pairing operation results in a link key that the two devices can use for link authentication and encryption after the pairing as well as for later reference. Each device stores the shared link key associated with the remote device address. The pairing process involves user interaction, e.g., entering a *pass key* (maximum 128 bits). For convenience, Bluetooth devices must be able to store a number of [*link key, device address*] pairs in a database. This opens up the possibility of link key compromise by a mobile virus as discussed in Section III-A. The pairing procedure itself consists of several steps [6]: initial key generation by the Host Controller Interface (HCI), generation and exchange of shared link key by the Link Management (LM), and baseband events. Programming errors and incorrect state machines in the Bluetooth stack can introduce exploits during the pairing procedure, as discussed in Section III-A. We discuss below potential vulnerabilities in Bluetooth that may be exploited by future mobile viruses.

A. Bluetooth Exploits for Mobile Malware

Figure 1 lists several vulnerabilities and exploits that have been reported for popular Bluetooth-enabled devices such as cell phones, headsets and Bluetooth software stack from vendors. Our goal is to provide a general classification of the vulnerabilities and therefore, we do not mention the specific manufacturer and model names that are affected by these vulnerabilities. The references cited in the last column of Figure 1 provide detailed information on each exploit and names of affected devices. Many of these exploits can be prevented by downloading the appropriate patches from the Internet. However, it should be noted that service providers and mobile phone vendors do not currently have any infrastructure in place to patch an existing Bluetooth software stack on phones when an exploit is discovered. They typically correct the software flaw in their future phones and model releases, since any recall of mobile phones is an expensive and time-consuming process. Therefore, malicious agents may take advantage of multiple unpatched vulnerabilities in an older Bluetooth stack. The risk can be partially mitigated by installing anti-virus software for mobile handsets. However, the anti-virus tools do not patch exploits in OS and Bluetooth software stack — they simply remove the infected files and directories from the handset. Therefore, it is crucial to develop *secure procedures* for on-demand and over-the-air (OTA) (or over-the-Internet) patching to protect against future mobile attacks. The various potential exploits and vulnerabilities are:

- ***Brute-force proximity scanning:*** There are brute-force techniques such as RedFang [11] that can be used to guess the Bluetooth device identifier (48-bit) of a remote device. The last 3 bytes of the identifier should be unique to each Bluetooth device, whereas the first 3 bytes of the address are specific to each manufacturer and are assigned by IEEE. In practice, some cellular phone manufacturers choose not to assign a unique Bluetooth identifier to each phone. By reducing the size of the searchable address space (e.g., when the manufacturer of the remote device is known, i.e., the device is within the visual range) and by improving the address search algorithm, it is now possible to find all hidden Bluetooth devices within range in about four and half days [11]. This means that the spreading rate of a mobile virus based on current-generation brute-force proximity scanning techniques is relatively low. However, further reduction in time is possible due to pseudo-random number generators used for addresses by device manufacturers, and by employing an approach similar to permutation scanning used by regular Internet worms. The brute-force approach is not a limiting factor for a malicious attacker who uses a Bluetooth-enabled device simply to launch a crossover worm or virus (discussed in Section II-A).
- ***Discovering addresses during communication:*** Since a Bluetooth device must be discoverable for pairing with a new device, its address can be recorded by a malicious user during the short time required to complete the pairing process. In many Bluetooth implementations, when a remote device is discovered, its address (device identifier) is not shown along with the device name. Since a device name can be easily forged by a malicious user, the remote user may think of it as a legitimate device in the network. A variation of this attack works very similarly to attacks setting up a fake public wireless LAN (WLAN) access point (AP) for enticing unsuspecting users. A malicious Bluetooth AP can accept connection requests from Bluetooth devices and record their addresses and paired link keys.
- ***Compromise of link key database:*** If a malicious agent gains root access on a Bluetooth device using a known exploit (e.g., the integer underflow exploit in Figure 1), it may be able to read the stored [link key, device address] pairs, effectively gaining access to a set of previously-paired devices. Finding victim hosts in this manner is

Vulnerability	Exploit	OS/BT Stack	Damages	Comments
"Helo Moto" attack on port 8	Incorrect state of "trusted device"	BT OBEX Push Profile and vcard	Attacker can take control of victim device via AT commands	Combination of BlueSnarf and BlueBug attacks [19]
"Blueline" attack	Buffer overflow	BT OBEX setpath()	Crashes cellular handset	Pre-attack pairing is required [11]
Pairing not needed for using a BT service profile	Malformed user interface when responding to "0x0d" or newline character	BT Voice Gateway user interface	Attacker can access phone book entries, SMS and other personal information via AT commands	Users can be enticed to grant connection via social engineering exploits [11]
L2CAP DoS Attack	Send multiple malformed (short) L2CAP packets to target device	BT L2CAP layer (l2cap.c in most BT implementations)	Denial of service (DoS) attack (device slows down and freezes after a short time).	Affects phones from multiple vendors [20,21]
OBEX Object Push buffer overflow OBEX File Share buffer overflow	Send a long filename (longer than 256 Bytes) in either ASCII or Unicode	OBEX Object Push application (Blue Neighbors.EXE) BT FTP client	Crashes device (memory stack is corrupted), remote code execution is possible via suitably formed packet	Affects devices deploying BT stack from selected vendor [22,23,24]
OBEX Object Push directory traversal	Can place a file anywhere on victim device regardless of user-specified path	ussp-push executable	Can place trojans in startup and system folders (e.g. worm/virus payload)	Requires victim to grant connection, no filename or path is presented at the time of connection request [11]
Remote audio eavesdropping	Null authentication and authorization values in Windows registry entries for BT headset and audio gateway	BT stack/driverfrom vendor	Allows attacker to remotely inject audio into a victim's speakers, as well as remotely monitor audio via the microphone	Requires special-purpose software to exploit the vulnerability [16]
Failure to handle exceptions	Malformed BT nickname in remote device	BT stack in Mobile OS (Symbian)	Denial of service (DoS) attack (device restarts when it discovers the remote nickname)	No known patches, affects a specific combination of Symbian and Nokia phones [18]
Failure to verify input string	Malformed remote device name request	HCI remote name request function call in Linux/BT implementation	Allows attacker to send any command string in place of device name, can take control of victim	Affected multiple Linux distributions [17]
Integer underflow	No check on sign of protocol value	Linux implementation of BT (Affix and Bluez)	Crashes BT, may allow root access via separate exploit	Affected multiple Linux distributions [15]

Fig. 1. Vulnerabilities and exploits in Bluetooth (BT)

very similar to an email or instant messaging (IM) worm obtaining addresses of potential victims from an address book or buddy lists on the compromised host.

- ***Software errors:*** Software and programming errors such as improper exception handling, buffer overflows, integer underflows, etc., can make the Bluetooth software stack highly vulnerable to remote attacks. We have highlighted several vulnerabilities [12], [13], [14], [15] of this type in Figure 1. For example, in one Bluetooth implementation, failure to check the received remote device name allowed an attacker to send any command string in place of the remote device name and therefore, take control of the device via a known root exploit. Software errors, in particular an application's failure to properly bound-check user-supplied input data, remain one of the most popular and effective exploits by Internet virus writers.
- ***Incorrect protocol handling:*** Incorrect protocol handling is another vulnerability that can be exploited by a mobile virus. The most notable vulnerabilities of this type are found in the Logical Link Control and Adaptation Protocol (L2CAP) layer [16], [17], [18] and Object Exchange (OBEX) protocol [19], [20], [21] of certain Bluetooth software stack (see Figure 1). These vulnerabilities can be exploited to launch DoS attacks or crash the target device.

IV. SMS/MMS Vulnerabilities

Unlike proximity scanning whose range is limited to a small area, mobile viruses that exploit Short (SMS) or Multimedia Messaging Service (MMS) to spread to other devices are capable of causing large-scale damage, similar to the scale of attacks seen on the Internet. This is because the structure of real-world SMS/MMS networks resembles a scale-free topology similar to email and IM networks, as we will see in Section VI. The primary reason why we have not witnessed any large-scale attacks involving SMS is a limit on the size of messages (up to 160 B) that can be sent via SMS. An MMS message does not have a size limit unless imposed by a service provider (typically up to 300 KB). While SMS messages are primarily restricted to text, MMS messages can have embedded text, audio (MP3, MIDI), images (JPEG, GIF) and video (MPEG). For example, the payload size of Commwarrior is 27 KB which can be easily sent to another phone as an MMS attachment.

Before discussing the vulnerabilities, a brief discussion of the SMS messaging system is in order. When a mobile user sends a message from a handset (i.e., Mobile Originated or MO) or a web-based gateway to another phone number, the message is received by the Base Station System (BSS) of the service provider. The BSS then forwards the message to the Mobile Switching Center (MSC). Upon receiving an MO message, the MSC sends the end-user information to the Visitor Location Register (VLR) of the cell and performs checks on the message for any violation. It then forwards the message to the Short Messaging Service Center (SMSC) of the provider. The SMSC stores the messages in a queue, records the transaction in the network billing system and sends a confirmation back to the MSC. The status of the message is changed from MO to Mobile Terminated (MT) at this point. Through a series of steps, the message is then forwarded by SMSC to the receiving user's MSC. The MSC receives the subscriber information from the VLR and finally forwards the message to the receiving handset. The store-forward nature of SMS makes it vulnerable to DoS attacks, as demonstrated by a recent study [22]. In what follows, we focus on vulnerabilities

that can be exploited by a future mobile virus or worm.

- ***SMS gateway errors:*** Many wireless providers allow Internet users to send short text messages directly to their mobile phone subscribers via a web-based SMS gateway. When not designed correctly, such a gateway opens the door to send large volumes of SMS spams and other malicious content. These gateways can benefit by using a visual challenge-response test such as CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) [23] to prevent automated spam generators from using the system. However, implementation errors (e.g., by not having enough characters in the CAPTCHA image) can make the SMS gateway vulnerable to brute-force attacks.
- ***Software errors:*** SMS/MMS systems can suffer from software vulnerabilities in embedded objects such as images or video. For example, a vulnerability in GIF image execution function [24] in MSN Messenger 6.2 could allow a remote attacker to execute arbitrary code caused by improper bounds checking of user-supplied input in the width and height fields of a GIF image file. MPEG stream buffer overflows due to incorrect bound-checking in MPEG players have also been reported — this can be exploited by invoking a vulnerable MPEG player within MMS. Similarly, a design flaw in the graphical user interface (GUI) of the Java API on a particular mobile phone allowed remote attackers to send unauthorized SMS messages by overlaying a confirmation message with a malicious message [25]. Successful exploits of software errors may allow a mobile virus to send unauthorized messages and execute arbitrary machine code within the context of the SMS/MMS application on the handset.
- ***SMS spoofing attacks:*** SMS address spoofing is an emerging threat that allows a malicious agent to make an SMS message appear as though it came from a different user and network. This is similar to how email spams and spam relays work. The SMS web-based gateways can also be exploited to spoof the message's origin. There appears to be even an open-source tool on the Internet called "SMS Spoof" for Palm OS that allows any user to send spoofed messages through any SMSC supporting the EMI/UCP protocol.

V. Agent-Based Malware Modeling

We now focus on mobile malware propagation. We briefly discuss an agent-based malware modeling (AMM) framework that we have developed to investigate malware exploiting SMS/MMS and Bluetooth vulnerabilities on cellular handsets. We refer to [26] for a detailed presentation of AMM and its various capabilities. Although computationally expensive, an agent-based modeling approach relaxes the homogeneity and perfect-mixing assumptions of standard epidemiological models by (i) incorporating heterogeneity in agent attributes (e.g., different mobile OSs, handsets), (ii) modeling the state transitions of an agent as an explicit stochastic process, and (iii) allowing highly-structured topologies of service interactions (e.g., SMS senders and receivers) among the agents. In AMM,

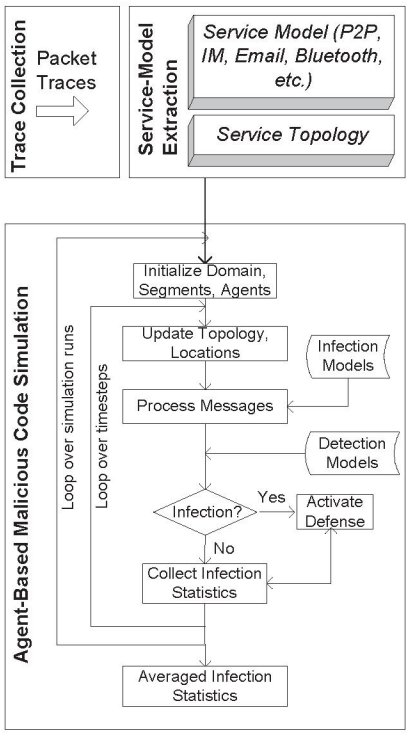

Fig. 2. Flowchart of AMM simulation steps

we model a mobile network as a collection of autonomous decision-making entities called *agents*. The agents represent networked devices within the network such as PDAs, mobile phones, service centers (e.g., SMS Center) and gateways. In case of agents representing mobile devices, the connectivity changes as users roam about the physical space of the network. The behaviors of the agents are specified by a set of services running on them. For example, an agent may consist of client programs for email and SMS/MMS messaging, whereas the SMSC agent may consist of a store-and-forward server only. Thus, there are two types of topologies in our simulation environment. The *physical* connectivity is determined by the physical network infrastructure, movement of the agents, location of access points and base stations, whereas the *logical* connectivity is determined by the messages exchanged among the agents. An agent may participate in multiple logical topologies corresponding to different services like email, IM, SMS, etc. We also group the agents in a hierarchical manner. For example, agents representing cellular base stations can keep track of mobile devices in their respective cells. Accordingly, these agents are able to collect information aggregated over the individual devices in their respective cells. This capability of higher-level agents to aggregate observations collected from lower-level agents reflects real-life processing of information within a mobile network. The information processed at these different levels can also be used to activate different response mechanisms against a spreading malware. Figure 2 shows a flowchart of our prototype simulator. The first step is to prepare the following input parameters for AMM: (i) *infection-model parameters* for the target services (Bluetooth and SMS for the present study), (ii) *topology* of service interactions among the agents (i.e., SMS messaging among the users), (iii) *location* of base stations, (iv) *mobility models* for devices that are mobile,

(v) *infection and replication state machine* (i.e., "attack vector") of malware, (vi) *detection model* of malware and (vii) *an attack response model* if any (containment, rate-limiting, anti-virus, etc.). At the beginning of a simulation run, agents are instantiated with appropriate class-specific data structures. At each timestep, the coordinates of mobile agents are updated based on their mobility models resulting in new physical and logical topologies. Next, each agent exchanges messages with other agents according to the SMS or Bluetooth service model — the probability of any of these messages being infected is calculated from the service-infection model. The time steps are repeated over a user-specified number of trials so that the results can be averaged over these trials. The simulator is general enough to experiment with different algorithms for malware detection and containment. The detection algorithm can be implemented at various levels of hierarchy, e.g., at individual devices, or at network operation centers, depending on the granularity of the detection algorithm. Similarly, when an infection is detected, containment actions, e.g., "detect-and-block" can be activated at various levels.

The SMS/MMS service parameters can be extracted from traces collected from a real-world cellular network, as we have used in our simulation results. The inclusion of a state diagram of the mobile virus, as described below, results in a more realistic epidemic spreading in AMM. An alternative is to simply input the topology of a particular service like SMS, and consider all nodes in the graph equally vulnerable to an SMS virus. The spread of the epidemic in this case solely depends on a constant infection probability and the topology of the SMS network. While most modeling literature on malware spreading that exploits specific services such as email or IM follows this methodology, this simulates only the worst-case attack scenario and may not represent the true spreading of the epidemic within the network.

Agent mobility: There are a variety of mobility models available to simulate different cellular and ad hoc wireless environments. We refer to [27], [28] for a discussion of these models. We have implemented two commonly-used models proposed for ad hoc wireless and cellular environments, namely, Random Waypoint (RWP) and Gauss-Markov (GM) mobility models, respectively.

In the RWP model, a node randomly chooses a destination in the simulation area and moves at a speed v chosen randomly from the uniform distribution $[v_{min}, v_{max}]$ along a straight path towards the destination. Then, the node pauses for a constant time t_{pause} before it chooses a new destination randomly. A node in the RWP model is, therefore, characterized by its current coordinates, speed, destination point and pause time. Following [29], we avoid the initial high variability in average neighbor numbers by discarding the initial 1000 seconds of simulation time and saving the mobile positions as the initial starting locations of our simulation.

The GM mobility model uses a Markov process in updating both speed and direction of a mobile node. Originally proposed for simulation of PCS networks [30], GM allows adaptation to different levels of randomness via a tunable parameter. In GM, each node updates its speed (s_t) and direction (d_t) at time t based on their values at time $t-1$ as:

$$s_t = \alpha s_{t-1} + (1-\alpha)\bar{s} + \sqrt{(1-\alpha^2)}s_{x_{t-1}} \quad (1)$$

$$d_t = \alpha d_{t-1} + (1-\alpha)\bar{d} + \sqrt{(1-\alpha^2)}d_{x_{t-1}} \quad (2)$$

where $0 \leq \alpha \leq 1$ is the tunable parameter, $\bar{s}$ and $\bar{d}$ are the mean value of speed and direction as $n \to \infty$, respectively. $s_{x_{t-1}}$ and $d_{x_{t-1}}$ are random variables drawn from a Gaussian distribution.

Bluetooth RF model: The connectivity of an ad hoc wireless network such as those formed by Bluetooth and other short-range RF devices strongly influences the effectiveness of malware spreading via proximity scanning. To determine if two Bluetooth-enabled devices are neighbors, one can simply use a threshold distance (r_0). For example, in case of Class-2 Bluetooth devices, $r_0 = 10m$. However, one should consider a more realistic wireless channel model by considering shadowing effects that are induced by the presence of obstacles. This means that the connectivity between two devices is now a stochastic parameter. Following the approach by Bettstetter and Hartmann [27], we adopt a log-normal shadow fading model to determine if an infected device can send a message to a nearby device using an existing vulnerability in the Bluetooth stack. In a shadow fading environment, the signal attenuation, $\beta(u,v)$ between a pair of nodes u and v is expressed as the sum of (i) a deterministic geometric component β_1 based on the relative distance $r(u,v)$ and (ii) a stochastic component β_2 where

$$\beta_1(u,v) = \alpha.10.log_{10}.(\frac{r(u,v)}{1m})dB \quad (3)$$

and β_2 is chosen from a log-normal probability density function:

$$f_{\beta_2}(\beta_2) = \frac{1}{\sqrt{2\pi}\sigma}.exp(-\frac{\beta_2^2}{2\sigma^2}) \quad (4)$$

where α is the path-loss component ($2 \leq \alpha \leq 5$) and σ is the standard deviation ($|\sigma| \leq 10dB$). For a given transmit power p_t and a threshold receive power $p_{r,th}$, two devices u and v are neighbors if the attenuation between them satisfies the inequality $\beta(u,v) \leq \beta_{th}$ where the threshold attenuation β_{th} is given by:

$$\beta_{th} = 10.log_{10}.(\frac{p_t}{p_{r,th}})dB \quad (5)$$

Eqs. (3) and (5) along with the mobility models provide us with a propagation model of a malware that exploits Bluetooth vulnerabilities and spreads to different areas of a mobile network as the users move about the physical space.

Next, we present the state machine of a generic mobile worm that we will use for our simulation studies.

Mobile worm state machine: In order to model a mobile malware in AMM, a state diagram of its infection and replication phases is needed. Figure 3 shows the state diagram of a *Commwarrior*-like mobile worm that we have implemented in AMM. The different state transitions taken by the mobile virus are shown along with code snippets using J2ME Wireless Toolkit API and Symbian C++ API. We have not presented

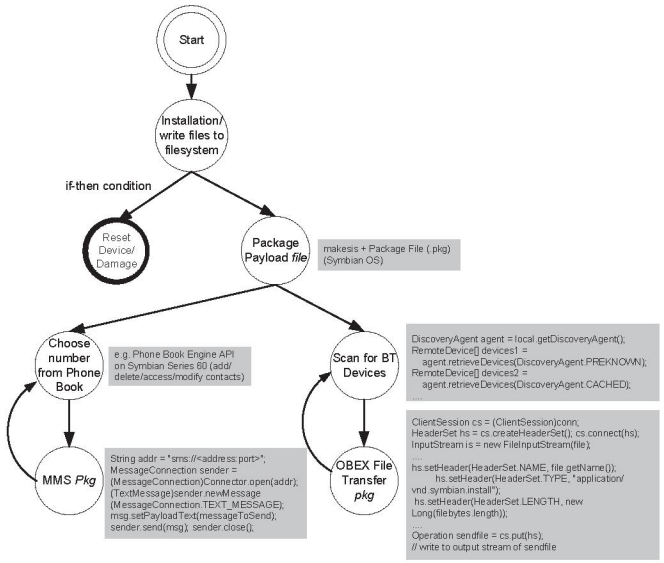

Fig. 3. State diagram of a mobile virus in AMM

the entire virus program although we doubt that a real-world mobile virus will be programmed in the manner we have shown. We implement virus state transition models in AMM in high-level languages (C++ and Java) for ease of programming. The first state transition takes place when a user accepts an incoming SMS message or a Bluetooth file exchange request. The process of acceptance allows the virus to write its executable and associated files (various .exe and .mdl files in case of Commwarrior) in the system and startup directories. Next, it prepares a payload to be sent to other vulnerable devices, a process called *packaging*. This state may make use of packaging applications already on the infected device, e.g., the *makesis* utility on Symbian phones. The third transition advances the virus to the *target acquisition state* in which it may use several propagation vectors, e.g., searching for a nearby Bluetooth device ("Scan for BT devices") or finding a contact address from the phone book on the device ("Choose number from Phone Book"). In case of Bluetooth, it may exploit device addresses already cached or pre-known (via the pairing process described in Section III) in addition to discovering nearby device addresses and their available service profiles. Access to the Phone Book can similarly be made via APIs provided by the device manufacturer or the OS (e.g., Symbian Series 60 Phone Book Engine API). The next state, *payload transfer*, comprises the actual process of sending the payload either via SMS/MMS or Bluetooth (e.g., using the Object Exchange or OBEX protocol). The target acquisition and payload transfer states can loop forever. Note that there is a second transition before the packaging state. Some viruses may test a condition, most commonly the current time and date, and cause damage to the handset. For example, Commwarrior.A (the initial version) resets the device if it is the first hour of the 14-th of any month.

The state diagram in Figure 3 and its implementation in AMM provide us with a realistic reconstruction of a real-world mobile virus exploiting Bluetooth and SMS, and study its propagation via simulation. We present results of our simulation next.

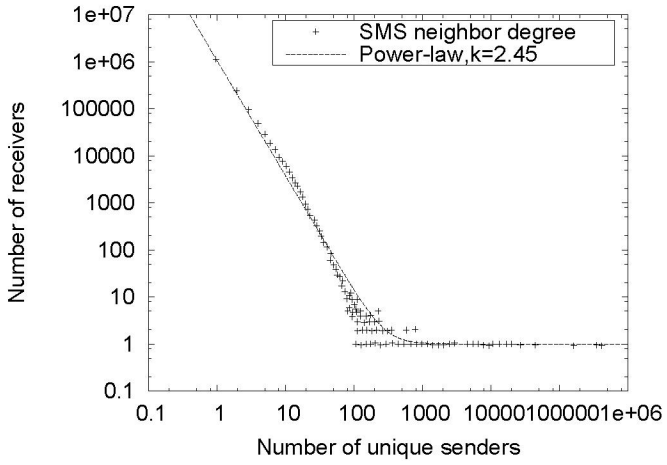

Fig. 4. Topological structure of the SMS network

VI. Simulation of a Mobile Worm

In this section, we present simulation results for propagation of a generic mobile malware developed in Section V and implemented in AMM. Our goal is to study the epidemic growth rate of a mobile worm in a realistic messaging network. To the best of our knowledge, there does not exist any malware propagation model using SMS/MMS services in a cellular network. A recent study [31] of SMS usage characterization collected call data records and SS7 traces over a three-week period from a large cellular carrier with 10 million mobile users. The data allowed us to reconstruct a realistic SMS messaging network with the following parameters: message sending rates, message receiving rates, cumulative density functions (CDF) of user-to-user message size (B) and message service times, and the SMS service topology. The original data involved a very large number of messages (over 59 million) and users (over 10 million). Therefore, we scaled the data to a small number of users and messages, while still preserving the basic characteristics of the original data sets. From the original number of exchanged messages in [31], we derived the total number of messages sent and received by the mobile users over our simulation time window of 3600 seconds. Figure 4 shows the topological structure of the SMS network by plotting the number of receivers for each unique sender. From the graph, it is clear that the real-world SMS network can be represented by a power-law network. We therefore fitted a power-law equation of the form $y = a + b \times x^{-k}$ to the data and calculated the power-law exponent, $k = 2.45$. This allowed us to scale the overall size of the network from over 10 million users to 2000 users while maintaining its scale-free properties. Figure 5 shows the CDF of message service times for delivered messages. The service time includes the queuing time at the SMSC and reflects the store-and-forward nature of the short messaging system.

Using the above data, we generated an SMS network in the form of an SMSC agent and 2000 mobile handsets in AMM. Table I presents the parameters for Bluetooth channel and RWP mobility models used in the simulation. We assumed 30% of the Bluetooth-enabled and 10% of the SMS-capable handsets immune from the worm. The initial number of infec-

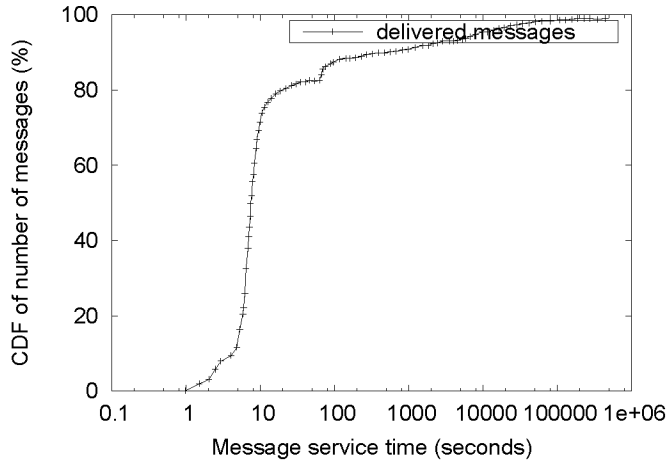

Fig. 5. CDF of message service times in seconds ([31])

Parameter	Value
α	3
σ	4 dB
β_{th}	30 dB
v (RWP)	[2,24]m/sec
t_{pause}	0
v	[0,0.1,0.9]
$I(0)$	[1]

TABLE I

PARAMETERS FOR BLUETOOTH CHANNEL AND MOBILITY MODELS

tions ($I(0)$) are 1 (Bluetooth) and 10 (SMS), respectively. Note that not all devices in our simulation have both Bluetooth and SMS services available. Figure 6 compares the average number of infections (averaged over 100 runs) for the worm exploiting SMS-only, and both Bluetooth and SMS. In order to simulate the heterogeneity in Bluetooth software stack, we introduced a Bluetooth vulnerability ratio (v) that specifies a fraction of the Bluetooth-enabled devices as susceptible to the worm. Figure 7 shows the effect of vulnerability ratio (v) on the overall growth rate of the worm, for $v = 0.1$ and $v = 0.9$. Figures 6-7 indicate that the growth rate of the mobile worm targeting SMS/MMS depends strongly on the messaging rates and the topology of the messaging network. If the initial infection of the worm is at a highly-connected node (e.g., an SMS stock quote server or a web-based SMS gateway), the spread will be very fast. However, since the vast majority of the SMS users have only a neighbor-degree of 1, the epidemic growth rate is small when the worm infects the average SMS user. On the other hand, when the users have a Bluetooth interface that is both vulnerable and discoverable, the mobility of the users have a strong effect on the mobile worm propagation. It effectively increases the neighbor-degree of an average user. In this case, a hybrid worm (SMS and Bluetooth) propagates much faster, utilizing both proximity scanning and the topological structure of the messaging network. While this is intuitive, our results reveal the exponential nature of the epidemic spread when both Bluetooth and SMS are exploited by the worm.

Next, we study the epidemic growth for different density levels of Bluetooth-equipped handsets in the network. To do this, we uniformly sample the handsets in the cellular network coverage area and choose a fraction of them to be Bluetooth-

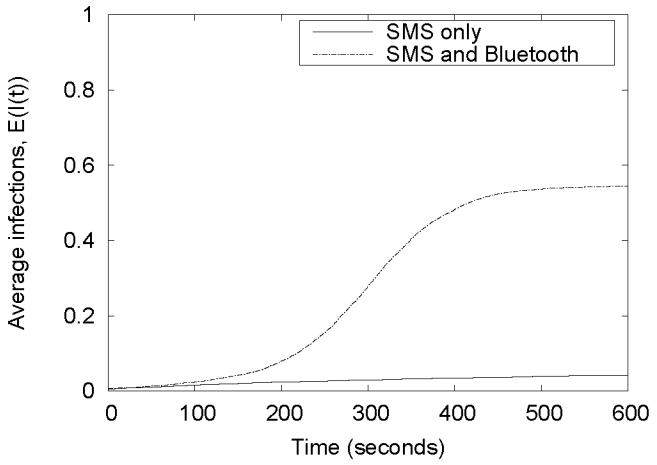

Fig. 6. Average number of infections for SMS and hybrid (SMS, Bluetooth) worm propagation

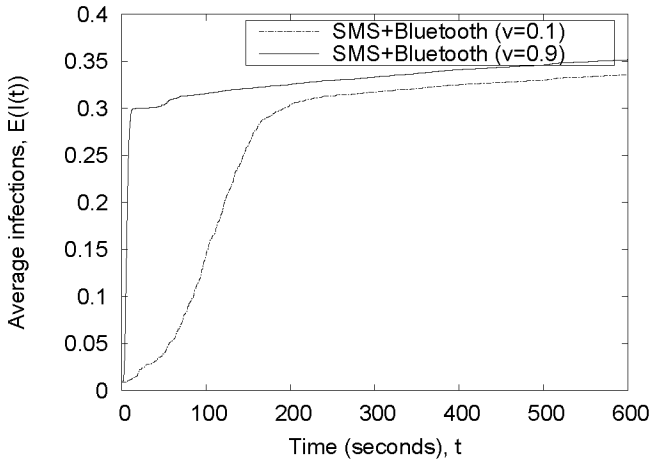

Fig. 7. Effect of multiple vectors (SMS and Bluetooth) on epidemic growth

equipped, resulting in a particular density of handsets per sq. ft. area. Next, we assume that only half of all handsets are vulnerable to the spreading worm. Figure 8 shows the epidemic growth rates for density levels of 0.3, 0.7 and 1.0. It is clear that when the density of Bluetooth devices is low, the incidence of the worm does not lead to an epidemic. This is related to the epidemic threshold of the cellular network. We also show the average fraction of infections at a moderate density of 0.7, for different vulnerability levels, in Figure 9. The results of Figures 8 and 9 are very similar — they suggest that a mobile worm epidemic can be effectively suppressed by either disabling the Bluetooth devices (Figure 8) or fast patching of the Bluetooth vulnerability (Figure 9), possibly via OTA updates.

VII. CONCLUSION

The proliferation of mobile devices such as smart phones, PDAs and handsets has introduced new mobile viruses such as Commwarrior, Mabir, etc., that can spread via SMS/MMS messages and by exploiting Bluetooth vulnerabilities. In this paper, we have analyzed the existing mobile viruses to extract a set of their common behavior vectors that can be used to develop mobile virus detection and containment algorithms. We have studied the vulnerabilities of Bluetooth

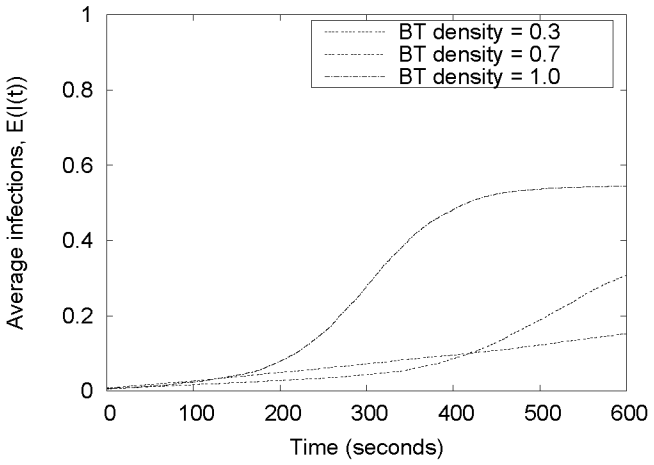

Fig. 8. Effect of Bluetooth handset density on epidemic growth

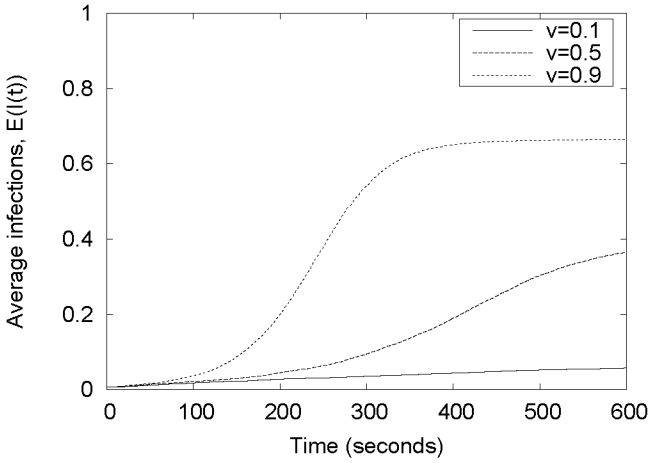

Fig. 9. Effect of vulnerability ratio in moderate density scenario

and SMS/MMS messaging systems in depth, and identified the vulnerabilities that may be exploited by future mobile viruses. We have also developed the state diagram of a generic mobile virus that can spread via SMS/MMS and Bluetooth. The discovery, infection and replication states of the generic virus were implemented in an agent-based malware modeling framework, called AMM, to study its propagation. We used data from a large real-world cellular carrier to generate a scaled-down topology of an SMS network and studied the propagation of the mobile virus. Our results indicate that due to heterogeneity of mobile handset platforms and scale-free nature of the SMS network, the growth rate of a mobile virus exploiting SMS messages is small. But the growth rate increases significantly when these handsets are highly vulnerable to Bluetooth exploits. There are several areas of future work such as improving our simulator so that larger topologies in the order of millions of handsets can be simulated, and investigation of novel mobile virus detection and containment algorithms.

Acknowledgement: The authors would like to thank the anonymous reviewers for their comments and suggestions for improving the paper. Scott Boehmer contributed to the development of the AMM simulator that was extended in the present work to study hybrid worms.

References

[1] Symantec, "Symbos.mabir worm description," http://securityresponse.symantec.com/avcenter/venc /data/symbos.mabir.html, April 2005.
[2] ——, "Symbos.commwarrior worm description," http://securityresponse.symantec.com/avcenter/venc /data/symbos.commwarrior.a.html, October 2005.
[3] In-Stat, "3g cellular deployment report," http://www.instat.com, March 2006.
[4] "Worldwide smart phone market," http://www.canalys.com/pr/2005/ r2005102.htm, 2005.
[5] RAV, "Vbs/timofonica virus description," http://www.ravantivirus.com/virus/showvirus.php?v=56, June 2000.
[6] "Bluetooth special interest group (sig)," http://www.bluetooth.com/Bluetooth/SIG/, 2006.
[7] 2nd International Workshop Networking with Ultra Wide Band, "Workshop on ultra wide band for sensor networks," July 2005.
[8] J. W. Mickens and B. D. Noble, "Modeling epidemic spreading in mobile environments," in *Proceedings of the 2005 ACM Workshop on Wireless Security (WiSe 2005)*, September 2005.
[9] "Common vulnerability and exposures database," http://www.cve.mitre.org, 2006.
[10] "National vulnerability database," http://nvd.nist.gov, 2006.
[11] O. Whitehouse, "Redfang 2.5," http://www.net-security.org/software.php?id=519, 2003.
[12] "Nokia affix bluetooth integer underflow," http://www.digitalmunition.com/ DMA[2005-0423a].txt, 2005.
[13] "Motorola bluetooth interface dialog spoofing vulnerability," http://www.securityfocus.com/bid/17190, 2006.
[14] "Nokia affix bluetooth btsrv poor use of popen(()," http://www.digitalmunition.com/DMA
[15] "Nokia symbian os malformed nickname vulnerability," http://securitytracker.com/alerts/2005/Mar/1013380.html, 2006.
[16] "Bluetooth helomoto attack," http://trifinite.org, 2005.
[17] "Nokia n70 l2cap dos attack," http://www.secuobs.com/news/ 15022006-nokia_n70.shtml#english, 2006.
[18] "Sony-ericsson bluetooth stack dos attack," http://marc.theaimsgroup.com, 2006.
[19] "Ambicom bluetooth object push overflow," http://www.digitalmunition.com/ DMA[2006-0115a].txt, 2006.
[20] "Nokia 3210 and 7610 remote obex denial of service vulnerability," http://www.securityfocus.com/bid/14948, 2005.
[21] "Nokia affix bluetooth btsrv/btobex vulnerability," http://www.digitalmunition.com/ DMA[2005-0712b].txt, 2005.
[22] W. Enck, P. Traynor, P. McDaniel, and T. L. Porta, "Exploiting open functionality in sms-capable cellular networks," in *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS)*, November 2005, pp. 393–404.
[23] L. von Ahn, M. Blum, and J. Langford, "Telling humans and computers apart automatically," 2004. [Online]. Available: citeseer.ist.psu.edu/vonahn04telling.html
[24] "Microsoft network messenger gif image code execution," http://xforce.iss.net/xforce/xfdb/19950, 2005.
[25] "Siemens s55 cellular telephone sms confirmation message bypass vulnerability," http://www.securityfocus.com/bid/10227, 2004.
[26] A. Bose, S. Boehmer, and K. G. Shin, "Malware spreading in mobile enterprise environments," The University of Michigan, Computer Science and Engineering Technical Report CSE-TR-520-06, June 2006.
[27] C. Bettstetter and C. Hartmann, "Connectivity of wireless multihop networks in a shadow fading environment," *ACM/Springer Wireless Networks*, vol. 11:5, pp. 571–579, September 2005.
[28] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," in *Wireless Communications and Mobile Computing*, vol. 2(5), 2002, pp. 483–502.
[29] W. Navidi, T. Camp, and N. Bauer, "Improving the accuracy of random waypoint simulations through steady-state initialization," in *Proceedings of the 15th International Conference on Modeling and Simulation*, March 2004, pp. 319–326.
[30] B. Liang and Z. Haas, "Predictive distance-based mobility management for pcs networks," in *Proceedings of the INFOCOM*, March 1999.
[31] V. Samanta, "A study of mobile messaging services," *UCLA Master's Thesis*, 2005.