SMART PHONE INTERACTION AND SECURITY

*[1]*Miss Priyanka J.Pursani*,
M.E 1st year,
priyankapursani@yahoo.com
priyankapursani@gmail.com
HVPM's COET,
Amravati.

[2]*Mr.P.L.Ramteke*
Associate Professor,
HVPM's COET,
Amravati.

ABSTRACT:

Internet has been reaching into every corner of the world and every aspect of our lives, providing us with anytime remote access and control over information, personal communications (e.g. through smart-phones) and our environment. So in this paper we present a system architecture that allows users to interact with systems located in their proximity using Smart Phones. These phones have the unique feature of incorporating short range wireless connectivity (e.g., Bluetooth) and Internet connectivity (e.g., GPRS) in the same personal mobile device. Due to these features, smart phones had been under the threat of virus, worms and malware which can cause dangerous attacks. So security of smart phones and protection to these threats is necessary.

KEYWORDS:

Software Architecture, Interaction Protocol, Malware

1. INTRODUCTION:

We believe that Smart Phones are the devices that have the greatest chance of successfully becoming universal remote controls for people to interact with various devices from their surrounding environment; they will also replace all the different items we currently carry in our pockets.

1.1 System Architecture

Our system architecture for universal interaction consists of a common Smart Phone software architecture and an interaction protocol.

Figure 1 shows the Smart Phone software architecture. In the following, we briefly describe the components of the software architecture.

•**Bluetooth Engine:** Bluetooth Engine is responsible for communicating with the Bluetooth-enabled devices .It is composed of sub-components for device discovery and sending/receiving data. The Bluetooth Engine is a layer above the Bluetooth stack and provides a convenient Java API for accessing the Bluetooth stack.

•**Internet Access Module:** Internet Access Module carries out the communication between the Smart Phone and various Internet servers. It provides a well-defined API that supports operations specific to our architecture (e.g., downloading an interface). The protocol of communication is HTTP on top of GPRS.

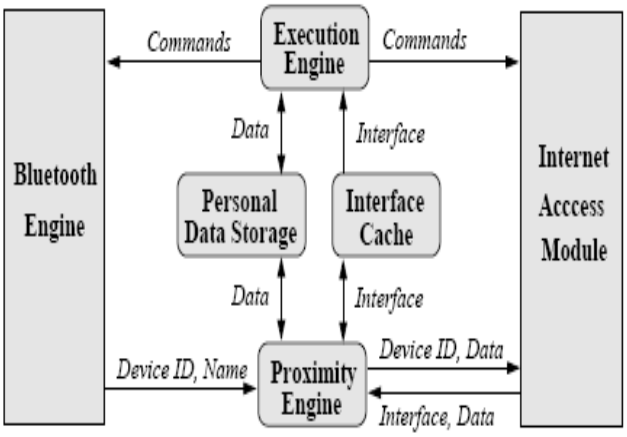

FIG 1.Software Architecture

•**Proximity Engine:** Proximity Engine is responsible for discovering the devices or systems located within the Bluetooth communication range. Each time the user wants to interact with one of these devices or systems, and an interface for this system is not available locally (i.e., a miss in the Interface Cache), the Proximity Engine is responsible from downloading such an interface. If the system has enough computing power and memory, the interface can be downloaded directly from it. Otherwise, the Proximity Engine invokes the Internet Access Module to connect to a web server and download the interface. The downloaded interface is stored in the Interface Cache for later reuse. Once this is done, the Proximity Engine informs the Execution Engine to dispatch the downloaded interface for execution. All further communication between the Smart Phone and the system happens as a result of executing this interface.

•**Execution Engine:** Execution Engine is invoked by the Proximity Engine and is responsible for dispatching interface programs for execution over the Java virtual machine. These programs interact with the Bluetooth Engine to communicate with the systems or with other Smart Phones . They may also interact with the Internet Access Module to communicate with Internet servers. For instance, the interface programs may need to contact a server for security related actions or to download necessary data in case of a miss in the Personal Data Storage.

•**Interface Cache:** Interface Cache stores the code of the downloaded interfaces. This cache avoids downloading an interface every time it is needed. An interface can be shared by an entire class of systems (e.g., Smart Locks, or Microwaves). Every

interface has an ID (which can be the ID of the embedded system or the class of systems it is associated with). This ID helps in recognizing the cached interface each time it needs to be looked up in the cache. Additionally, each interface has an associated access handler that is executed before any subsequent execution of the interface.

•**Personal Data Storage:** Personal Data Storage acts as a cache for "active data", similar to Active Cache. It stores data that needs to be used during the interactions with various systems. Examples of such data include digital door keys and electronic cash. Each data item stored in this cache has three associated handlers: access handler, miss handler, and eviction handler. Each time an interface needs some data, it checks the Personal Data Storage. If the data is available locally (i.e., hit), the access handler is executed, and the program goes ahead.

Figure 2 shows the interaction protocol that takes place when a Smart Phone needs to interact with an embedded system. We consider that any embedded system is registered with a trusted web server (this web server can be physically distributed on multiple computers). At registration, the web server assigns a unique ID and a URL to the device. All the information necessary to interact with the device along with a user interface is stored at that URL. This URL may be common for an entire class of systems. The user invokes the Proximity Engine each time she needs to interact with a device located in the proximity. Once the systems in the proximity have been identified, the user can choose the one she wants to interact with. Consequently, a request is sent to the embedded system to provide its ID and URL. Upon receiving the ID and URL of the embedded system, the Smart Phone executes the access control handler, and then, loads and executes the interface. In case of a miss in the interface Cache, the interface needs to be downloaded on the phone either from the web server or from the embedded system itself.

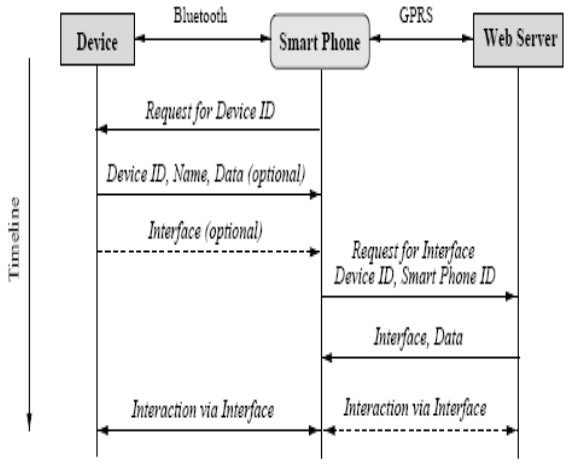

Figure 2.Smart Phone Interaction Protocol

2. SECURITY:

2.1 Attack vectors for mobile malware:

• **Bluetooth:** Many mobile devices have the capability to communicate with other devices in a short range using the Bluetooth technology. However, several flaws exist in both the protocol as well as its implementation. Some mobile malware exploit these to spread. Others disguise themselves as legitimate applications (Trojans) and try to spread to other devices that are within its Bluetooth communication range.The first known mobile malware, Cabir spread through Bluetooth. Malware that spread through Bluetooth can only communicate within the range of communication of Bluetooth devices (typically a few meters).

• **SMS, MMS, WiFi:** Some mobile malware spread themselves through SMS, MMS or WiFi technology. Most of these send SMS or MMS to other phones and attach themselves to the message that they send. ComWar, for example, spreads through MMS. There also exists a buffer overflow vulnerability in the SMIL (Synchronized Multimedia Integration Language) parser on mobile devices based on the Microsoft Windows Mobile 2003 operating system. This parser is used for parsing incoming MMS messages.

• **Vulnerabilities in the operating system:** Vulnerabilities exist in the operating systems used by mobile devices. Symbian OS, included as the operating system in most Nokia mobile phones, has several vulnerabilities. For example, one vulnerability found in the Symbian Series 6.x devices (Nokia 3650 and Siemens SX-1) is to create a file called "INFO.wmlc" in the root folder with 67 spaces between the "INFO" and the ".". This causes the mobile to work slowly or even crash.

2.2. Protection and Prevention Mechanisms against malware:

• **Keeping the device in non-discoverable Bluetooth mode:** Since leaving a Bluetooth-enabled mobile device in discoverable mode makes it vulnerable to attacks by mobile malware and hackers that exploit the documented vulnerabilities in Bluetooth, it is best to turn off the Bluetooth discovery mode on the mobile device.

• **Filtering out malware at service provider:** MMS messages that carry malicious payload can be detected at the service provider based on their signatures and thus can be filtered out at the service provider itself.

•**OS hardening:** Smart-phone OS can enforce some security features, such as always displaying callee's number and

lighting up LCD display when dialing. This can be achieved by only exporting security enhanced APIs to applications. With hardened OS, unless attackers can subvert the smart-phone OS without being noticed, attacking actions from malicious user-level code can be more easily detected by the smart-phone user.

3. CONCLUSION:

Thus the universal interaction architecture presented is the dual connectivity feature of Smart Phones, which allows them to interact with the close-by environment through short-range wireless networking and with the rest of the world through the Internet over cellular links.

Due to the interaction, it is easy to visualize that in the near future, the threat posed by mobile worms and viruses can cause considerable harm to the users of such devices and hence protection is necessary.

REFERENCES:

[1] Smart Phone: An Embedded System for Universal Interaction www.paul.rutgers.edu/~nravi/smartphone.pdf

[2] Mobile Worms and Viruses. www.it.iitb.ac.in/~manojk/MobileWormsAndViruses.pdf

[3] Smart-Phone Attacks and Defenses. www.research.microsoft.com/~helenw/papers/smartphone.pdf

[4] Helen J. Wang, Chuanxiong Guo, Daniel. R. Simon and Alf Zugenmaier. Shield: vulnerability-driven network filters for preventing known vulnerability Exploits. In *Proc. SIGCOMM,* August 2004.

[5] MIDP Profile. http://wireless.java.sun.com/midp/.

[6] General Packet Radio Service (GPRS). http://www.gsmworld.com/technology/gprs/intro.shtml

[7] P. Cao, J. Zhang, and K. Beach. Active Cache: Caching Dynamic Contents on the Web. In *Proceedings of IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*