DATE

Page No 24

MOBILE MALWARE

- New-generation smartphone combine the functionality of a cellphone and low-end PC.

- They are used for storing confidential doc, communicating via e-mail / SMS / MMS and taking photographs.

- They support feature-rich applications that run on top of a complete OS.

- The most common OS on smartphones is the Symbian followed by Windows Mobile, Linux and the Mac OS X. They provide a rich set of APIs to access the phone book and other files, send SMS / MMS msg, etc. These very APIs can also be used by malware to, eg:- read a confidential document on the smartphone and ship it to the attacker as an MMS attachement.

- In case of Symbian OS users can download new application onto their smartphones. New applications as well as updates of existing applications are packaged in symbian Installation Source (SIS) files. The sometimes when updations are made it invokes the malware code.

RAMSONS

- Bluetooth supports a host of tasks such as the exchange of e-business cards b/w two smartphones users, communication with point-of-sale terminals, headset to smartphone communication and "sync"ing b/w smartphones and a PC.
- Bluetooth has turned out to be an important vector of mobile worm propagation.

Bluetooth

- Bluetooth is both a communication technology and a protocol stack. It supports short-range wireless communication - a max b/w 10 and 100 m. b/w devices.
- Like Wi-Fi, Bluetooth uses 2.4 GHz Shortwave radio technology. Power req are lower than Wi-Fi. Transmission speed is limited to 5Mbps.
- Bluetooth is a complex, multi-layered protocol.
- The implementation of Bluetooth on Linux systems exceeds 25,000 lines of kernel code.
- Buffer overflow, integer underflow, and other vulnerabilities

have been explained.

- Exchange of files using Bluetooth

step1 The device broadcast an inquiry request to find other Bluetooth-enabled devices.

step2 All devices in the range of the initiator that are in discoverable mode respond sending their bluetooth device address (BD_ADDR). This is a 48-bit MAC address - the first 24 bits identify the device manufacturer/model and the last 24 bits specify a particular instance of that model.

- A device 'A' can set up a connection to any other device 'B'. But to set up such a connection 'A' should known the BD_ADDR of 'B'. One way of obtaining B's BD-ADR is through discovery procedure

- Knowing B's BD_ADDR, A could exchange files using the OBEX (Object exchange) protocol. OBEX is used to transfer images, business cards, and other files b/w Bluetooth devices.

- An attacker 'A', could use OBEX Push to transfer a file containing malicious code to user B,
- User authorization is usually required before a file can be accepted by the smartphone. Each user selects a PIN which varies b/w 4 and 16 characters long (usually 4 characters). The smart phone usually prompts a user to enter his/her PIN as a way to confirm whether an external file should be accepted.
- Hence keeping a smartphone in discoverable mode, a user risks revealing his smartphone's BD_ADDR to an attacker. By disabling authorization or by careless acceptance of external files/ attachements, a user keep his smartphone open to Trojans & other malware.

Link level Security

- Other than user authorization there is another level of security provided by the Bluetooth called the link-level authentication & encryption. ∴ both the side computes a common secret called the linkkey

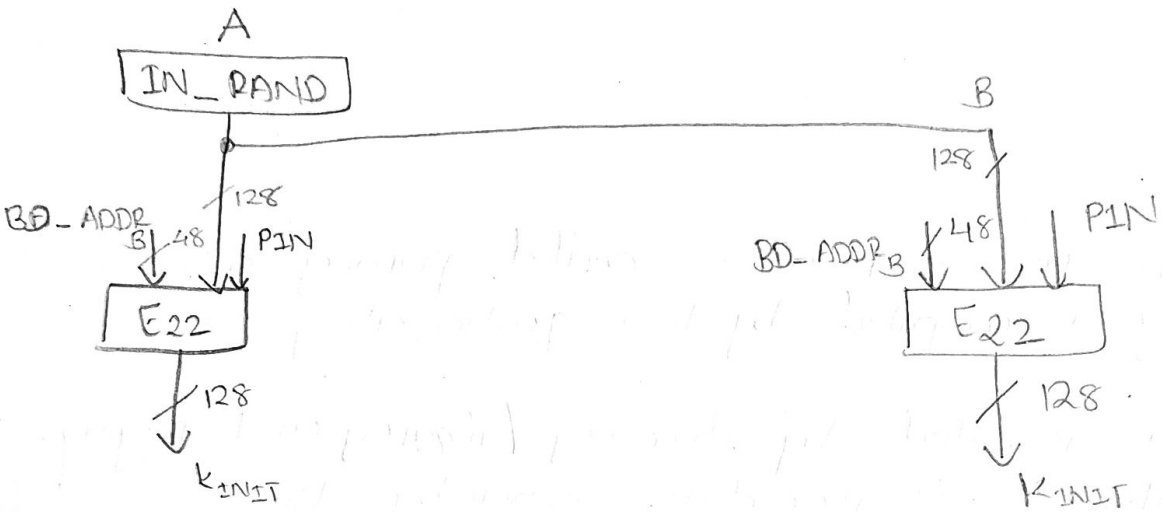
Bluetooth uses a procedure called pairing wherein this key is computed by two participating devices

Pairing is preceded by discovery / inquiry and paging. The latter is a procedure whereby the discovering device, 'A', establishes a connection with the discovered device, 'B'.

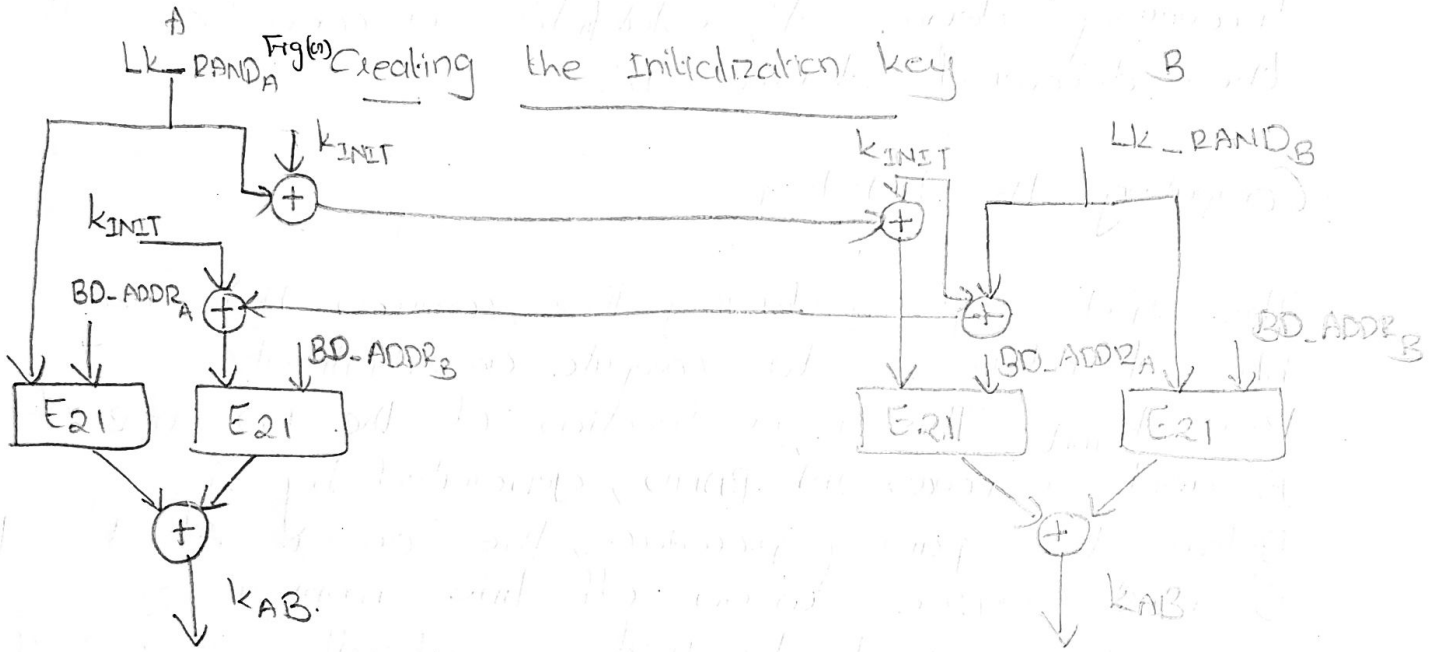
Computing the Link key

The first step in deriving the common link key b/w A and B is to compute an initialization key, K_{init}. This is a function of the BD_ADDR of B and a nonce, IN_RAND, generated by 'A'. Before the pairing procedure, the owners of 'A' and 'B' must agree in an off-line manner on a temporary PIN to be used specifically as part of the pairing procedure. Both users then type in the temporary PIN. ~~K_{init}~~ is also ~~a function of this temporary~~. K_{init} is computed from IN_RAND, BD_ADDR_B and the PIN using an alg, E_{22}, based on a block cipher called SAFER +.

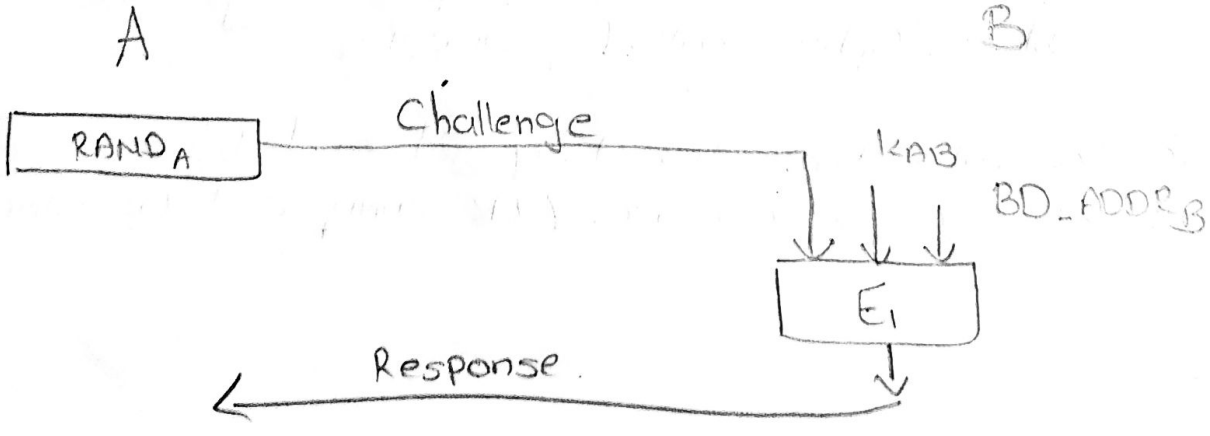
To compute the link key, 'A' and 'B' each generate a random NO. (LK_RAND_A and LK_RAND_B)



Fig(a) Creating the Initialization key



Fig(b) Creating the link key



Fig(c) Authentication

Each party then performs an XOR of its random no. with K_{init} and they transmit it across. Each side recovers the other party's random No. by performing an XOR of the received value with K_{init} fig(b). Now, each side has LK_RAND_A, LK_RAND_B and the two device addresses, BD_ADDR_A and BD_ADDR_B. They then perform identical operations on these to obtain the link key, K_{AB} as in fig (b). The operations involve the use of an alg E_{21} which like E_{22} is based on the cipher, SAFER +.

Thereafter, each device stores the pair, BD_ADDR of the other device and the newly computed link key in a DB. Each device maintains such DB of BD_ADDR, link key pair, one pair per device it is paired up with.

Using the Link key

The link key is used for encryption and authentication.

Suppose A and B were already paired, Let K_{AB} be their common link key. Now suppose

A wishes to authenticate B. For this purpose, it generates a random no., $RAND_A$ (a challenge) and sends it to B. The response computed by B is $E_1(K_{AB}, RAND_A, BD\_ADDR_B)$. Here again, E_1 is a function that is based on the block cipher, SAFER +. A also computes $E_1(K_{AB}, RAND_A, BD\_ADDR_B)$ since it has K_{AB}. It can then verify whether the response from B tallies with what it computed.

Hacking the Link key

- It is possible to launch a dictionary attack by sniffing each msg involved in pairing and authentication. These attacks enable an eavesdropper to obtain the link key, K_{AB} making it possible for the attacker to impersonate B to A. The latest version of Bluetooth — version 2.1 gets rid of this problem by using Elliptic Curve Diffie-Hellman (ECDH) key exchange.

Examples

Cabir was one of the earliest proof-of concept worms that targeted the Symbian series 60 OS. in 2004, it was authored by the International Virus writing group 29A. The worm attempts to discover other Bluetooth-enabled phone set in discoverable mode. When it finds such a phone, it sends the worm payload in a SIS file. The receiver needs to accept and install the file. Its payload was mostly benign typically displaying "Caribe" on the screen. However, the continuous scanning for new victims by an infected phone depletes battery power.

Commwarrior, which appeared in March 2005, was the first worm to spread through both Bluetooth and MMS. Like Cabir, it targeted Symbian smartphones. It used MMS to spread to different contacts in the smartphone's addr book. It requires user interaction to be installed. It entices the user with catchy subject headers such as "Happy Birthday" etc. Once it infects a smartphone, it attempts to discover Bluetooth-enabled smartphones and pass on the

infection as a SIS file to them.

While Symbian phones have been the most widely targeted, Windows and J2ME (Java) phones have also been the target of attack.

- WinCE.Duts was one of the first worms to target the Windows CE OS while the Redbrowser trojan was the first to target J2ME phones. Finally proof-of-concept "crossover" worms have also been observed. These are worms that spread from a PC to a smartphone and vice versa when the two are being "synced".