Four Malware and a Funeral

Axelle Apvrille (aapvrille@fortinet.com)*
Jie Zhang (jzhang@fortinet.com)†

Abstract: This paper selects four malware targeting mobile phone platforms, namely Eeki, Yxes, Redoc and GameSat. They are currently among the most relevant malware in terms of prevalence, or because they are precursors of their kinds.
A technical description is provided for each: how it infects the phone, its malicious payload and propagation mechanism. The descriptions in this paper are mostly new.
The paper concludes on two common trends: the simplicity of such malware - only little skills are required to implement them - and their monetization. There are so many mobile phone users that malware authors have the opportunity to get quickly rich. So, in the end, the *funeral* is for the user's bank account...

Keywords: malware, mobile phone, reverse engineering, SMS.

1 Introduction

Malware for mobile phones are often regarded as a very rare disease, found in research labs by a few weird-looking techies. It is true that, with only 450 different mobile species, they are clearly outnumbered by PC malware. The risks have however proved not to be that insignificant because few mobile species *does not mean few infections*. For instance, the costs and inconveniences caused by the famous CommWarrior worm, with 115,000 infected phones [Gos08] and over 450,000 messages sent [Hyp07] prove the matter cannot be underestimated.

It appears that 2009 has actually been a busy year for mobile malware, with several new malware families out during that year. We saw the first worms for iPhone in the wild, multiple versions of an advanced botnet-like worm for Symbian and dozens of malicious midlets sending SMS messages to premium numbers or spyware listening into conversations or SMS/MMS. Those malware regularly hit the headlines [Dan09, McM09] however once the turmoil is over, they quickly lose interest for the masses. It is consequently quite rare to find articles describing the inner mechanisms of those nuisances, i.e what they do, how they do it and how we found out. This is what this paper wishes to address.

The paper is organized as follows. First, we explain how we have selected representative malware candidates for 2009. Second, we briefly provide references to related work concerning those malware. This also highlights the contributions of the paper. Then, unlike *"Four Weddings and a Funeral"* [Cur94], the next sections detail four unhappy events: the four malware we selected for 2009. As we see in the last section, those four malware unfortunately lead to the funeral of our bank accounts, a fact we discuss among other upcoming trends.

* Fortinet, 120 rue Albert Caquot, 06410 Sophia Antipolis, France
† Fortinet, North 5 Floor, Software Tower 80, 4th Avenue, Economy Technology Development Zone, Tianjin 300457, China

2 Selecting Representative Malware

Mobile phone malware have different malicious intents: locking the phone, sending SMS messages, stealing contacts etc. In some cases, some malware even convey several of those malicious deeds. Below, we therefore classify them according to what seems to be their *most important* goal:

- **Annoywares** (approx. 35%). Once on the mobile phone, they make end-user's life difficult, disabling applications, changing fonts or causing phone's reboot.
- **Spywares** (30%). Those are also called *Trojan Spywares*. They target the end-user's privacy. They record voice calls or the phone's surroundings (microphone bug), get a copy of all SMS, MMS, e-mails and contacts, take screenshots of the phone, activate the camera etc.
- **Monetized malware** (20%). Their main goal is to make *money*, for instance by having the end-user call premium numbers.
- **Worms** (10%). Those malware do not have any clear intent apart from propagating to other devices using one or several communication means of mobile phones: Bluetooth, Wifi, Http, SMS, MMS, USB... Many Proof of Concept (PoC) malware fall in this category, but some of them are not PoCs, such as the famous CommWarrior, or 2009's Yxes that we will discuss in Section 5.
- **Hacking tools** (5%). Specific tools meant to circumvent a given limitation, e.g install non-signed applications, or help discover vulnerabilities and build malware.

In 2009, we noticed a strong increase in monetized malware and spywares. As spyware are often borderline malware, for instance because they are commercialized as parental locks, some people consider they are legit while others ban them. We consequently preferred to select two malware from the monetized category, so that their maliciousness could not be disputed. We selected them on two different mobile platforms: GameSat, a Java midlet (Java is supported by nearly all mobile phones) and Redoc, a Windows Mobile malware.

Then, we selected the remaining two malware for their fame: two worms, one for Symbian (Yxes) and one for the iPhone (Eeki). Both of them made the IT headline for several weeks - Yxes because it was close to a botnet, and Eeki as the first worm for iPhone - so we felt a detailed analysis would arouse more interest.

Note that we did not select any '*annoyware*' nor hacking tools because we only encountered few of them last year. Indeed, similarly to PC's world, malware authors do not any longer seem interested in locking the device or causing malfunction. As for hacking tools, there are only few altogether.

Finally, with GameSat, Yxes, Eeki and Redoc, note that all major mobile platforms are represented, apart from BlackBerry phones who have not much been targeted yet.

3 Related Work

In research conferences, the best known papers concerning mobile phone malware are probably the work of [Hyp07] or [MM09]. The former provides interesting background

information, trends and status for mobile malware. It is now 3 years old and would be worth a few updates - which this paper shall try to provide between lines. The latter details advanced hacks to discover vulnerabilities on new smartphones. It is interesting for security research but does not focus on malware which are most in the wild currently, as those we selected.

Rather, for the analysis of our malware short list, this paper uses reverse engineering tutorials, virus encyclopedia descriptions or informal technical blog entries. Eeki is particularly well-documented in the long technical report of [PSY09]. Our paper only tries to present the worm in a shorter and more educational way. We also highlight the re-use of a non-malicious program, a fact we found during our reverse engineering and missed by the report.

For Symbian's Yxes, the reading of the massive primer on reversing Symbian applications [SN07] or [Zha07] is particularly valuable. Both help the anti-virus analyst reverse Symbian applications. More specific information is also taken from our previous work [Apv09b, For09b]. Concerning Yxes, this paper is the first to explain the worm's mechanism from end to end.

As for GameSat and Redoc, we are not aware of any previous work concerning them, apart from virus encyclopedia descriptions such as [For09a, For09c] or other anti-virus companies. Yet, those viruses have several siblings and they are currently in the wild.

4 iPhoneOS/Eeki.B!worm (aka Ikee)

4.1 Infection

In November 2009, several malware targeted the iPhone (a Proof of Concept Python script, a ransomware and two worms). This section concerns the last one to be released and the most advanced, Eeki.B.

All these malware rely on the same security hole. Among other things, jailbreaking an iPhone installs an OpenSSH server on the iPhone. This consequently opens up access to the iPhone to whomever knows a valid login/password. Unfortunately, it turns up that all iPhones are shipped with a default root account whose password has publicly been reverse engineered to 'alpine'. Consequently, if the default password isn't changed immediately after jailbreaking, anybody can log on to the phone via SSH, as root, using the default password. So, the malware need only try and connect to the SSH port with those credentials. If it is successful, they are in (and root) to execute their malicious payload.

4.2 Payload

This subsection describes the payload of Eeki. Figure 1 illustrates the various steps.

On the phone, Eeki mainly consists of an installation script (in charge of installing the malware on the device), an infection daemon to propagate and infect other devices and a payload daemon achieving the malicious actions.

The installation script (named *inst*) is the first Bourne shell script to be run on the infected device. It basically achieves the following tasks:

- Configure the system so that the infection daemon runs automatically after reboot. On iPhoneOS, this consists in setting an XML property file for the daemon, and

Fig. 1: Workflow of iPhone worm Eeki.B. The *inst* script configures the phone, sends information to a remote server, and spawns the payload daemon, *syslog*. The payload daemon contacts a remote server, downloads a malicious script and runs it on the infected device. After reboot, the infection daemon, *sshd*, scans for new victims, zips the malware and propagates to new victims.

specifying the *RunAtLoad* option. Then, start the daemon, using iPhone's *launchctl* command. Note it is named *sshd* so as not to arouse suspicions, but it is absolutely not an SSH daemon.

- Download and install necessary packages on the iPhone, such as CURL or SQlite support.
- Configure the system so that the real payload daemon runs automatically every 300 seconds (*StartInterval* option). Then, start the daemon. Same as the infection daemon, it is illegitimately named *syslog*, but has no relationship with a syslog daemon.

The payload daemon is another Bourne shell script. It contacts a remote malicious URL by HTTP, and downloads another malicious script from that location. As the program sending the HTTP requests had been renamed *duh* by the malware authors, analysts initially thought it was malicious. In some of our previous work [Apv09a], by the study of its assembly code, we found out it was only the port of the legitimate htmlget program. This program sends an HTTP GET request (in that case to a malicious remote web site) and waits for the response. So, duh is not malicious, but used by the payload daemon to download a malicious script.

The remote malicious URL was quickly shut down, but F-Secure researchers got their hands on the malicious script to download (before the website was closed). They found

out that it both redirected the online ING Direct's website to a phishing website, and also tried to steal TAN codes (TAN codes are authentication codes commonly used by banking websites) from the victim's iPhone SMS database. The intent is clear. Note that downloading a script from a remote URL also helps the malware authors make their worm evolve according to their malicious intents. Had the website not been taken down, they could have moved to target another bank.

4.3 Propagation

When launched, the infection daemon goes in search of other victims to infect. The algorithm is extremely simple:

1. Scan networks for iPhones with an SSH port onto which credentials root/alpine work. In particular, Eeki.B scans IP ranges belonging to T-Mobile, Vodafone, Optus, Mobilkom, Pannon GSM Telecom. The daemon tests whether another iPhone is vulnerable by simply trying the following:

   ```
   sshpass -p alpine ssh -o StrictHostKeyChecking=no root@host 'echo 99'
   ```

   The daemon checks that '99' is returned. If so, it means the SSH login was successful, and that the target is vulnerable.

2. Infect the vulnerable target. First, the infection daemon copies an infected package (a .tgz which contains a copy of all necessary files: daemons, property files, scripts) via SCP to the victim. Then, the script remotely launches (via SSH) commands to untar the package and run the installation script:

   ```
   cd /private/var/mobile/home/;tar xzf cydia.tgz;./inst
   ```

The infection daemon is an executable, so all this information was reversed using a disassembler such as IDA Pro. Additionally, the *strings* utility on Linux is very helpful to spot readable text strings within the executable. The remote infection commands were found that way.

5 SymbOS/Yxes!worm

Yxes is another famous worm of 2009, with significantly advanced features: encoded malicious URLs included at the end of the Symbian packages, automated and silent installation of new malware, silent access to remote web sites etc. This section describes how the malware works.

5.1 Primo Infection

Initially, the victim gets infected by installing the malware from a download server (controlled or infected by the malware authors). Its 'attractive' package name (sexySpace.sisx, beauty.sisx, sexy.sisx...) are perhaps reasons why the worm has initially spread.

In an other case, the worm has been reported to trojan a legitimate application [Cyb09]: the victim thinks he/she is installing the legitimate application and does not know the

package also includes a malware. Those are primo-infection cases. They usually consists of a variant A, B, D or E of Yxes (see Figure 2).

Subsequently to primo-infection, infected phones contaminate other phones by sending them SMS messages which include a link to one of the malicious web servers controlled by the malware author(s).

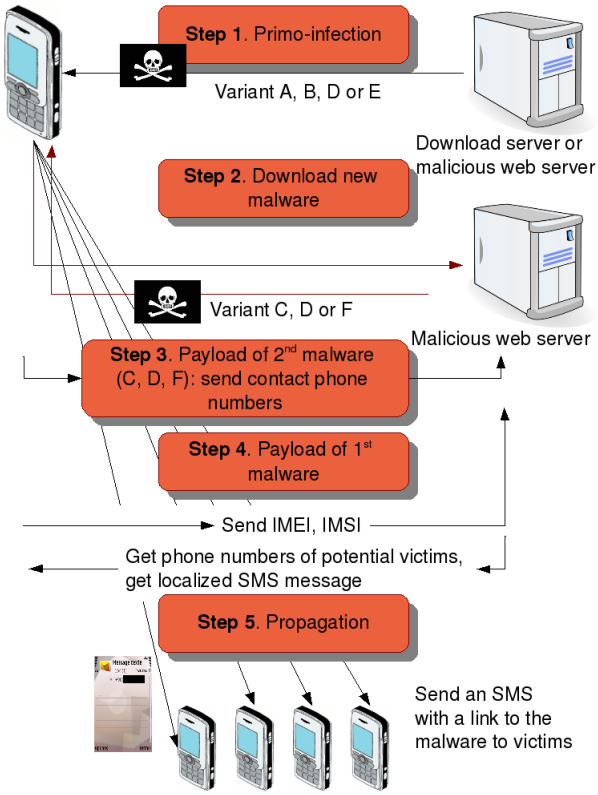

Fig. 2: Global overview of Yxes's interactions with remote servers

5.2 Malicious Payload

Upon installation, the malware decodes the malicious remote servers' host names (www.mozi11a.com, www.megac1jck.com etc) with a simple XOR loop and then tries to contact them (HTTP request).

On the server's side, the HTTP request is processed by Java Server Pages, and the server replies with a newer update of Yxes or another variant (usually C, D, F or G). The server scripts make sure to select a malware which is compatible with the victim's phone, to ensure better propagation. This new malware is silently installed on the victim's

phone, using the RSWInstSilentLauncher class of Symbian's SW Installer Launcher API. This results into another infection of the victim's phone (over-infection).

If a variant C, D or F is downloaded, the variant sends phone numbers harvested on the infected phone to a specific Java Server Page on the malicious servers. More precisely, variants C and F parse the victim's contacts and send all phone numbers[1]. whereas variant D only sends its own phone number.

In parallel, the initial malware (and the new one) runs in background its malicious activities such as killing specific applications (File managers, task managers...) to make reverse engineering more difficult to analysts.

It also sends to the remote servers more information on the victim: phone's model, IMEI, IMSI, malware's version number. Some pages of the remote servers are configured to respond with a different (encrypted) payload, depending on the Mobile Country Code (MCC) of the victim's phone. Although the encrypted payload hasn't been reversed yet, a sensible guess is that this file contains phone numbers of other potential victims in the same country. This guess is backed up by the fact the server's page is named 'NumberFile.jsp' and by the fact our test phones created SMS drafts for French phone numbers and Saudi Arabia, whereas screenshots of Yxes in China clearly show the malware only contacts other numbers in China. In that case, the servers also help to control malware's propagation. The authors can target a given country or even conduct a DDoS on a specific phone number. Propagation control is also extremely handy if the malware is in a debugging phase.

The Yxes worm also contacts a script named TipFile.jsp, which could contain the SMS text. This guess is backed up by the fact the malicious script takes a LanguageCode parameter. This would be perfectly appropriate to customize the text's language. Obviously, Chinese victims are more likely to follow a link inside an SMS written in Chinese, English victims an SMS written in English etc.

5.3 Propagation

Once the phone numbers of future victims and SMS text have been downloaded, the malware begins its propagation phase: it *silently* sends (i.e without user's consent) an SMS to each new victim (see Figure 3), with the customized text, and a link to a malicious server where the victim can download the malware. SMS cannot include any attachments, so the malware cannot attach itself to the message. Instead, it has to rely on an additional entity, the malicious servers, for its propagation. Hence, this is an indirect propagation. The malware author(s) could have used an MMS to spread the malware as they may include attachments. However, fewer phones are configured to receive or send MMS (only 40% in France according to [Oci]), contrary to SMS which are so popular. It is quite possible SMS is a better propagation vector after all.

It should also be noted that Yxes does not replicate on the victim's mobile phone. A few strings such as c:\kel.sisx, c:\root.sisx, spotted in the malware's binary, initially mislead analysis and people thought the malware copied itself in those files and then spread (on the memory card or via HTTP). This is actually wrong. A close reverse engineering of those routines have shown the malware does not (currently) copy itself in those files nor

[1] For variant C, this behaviour is a (sensible) guess, not a proof, because the exact parts that send the contacts haven't been identified. Code parsing and storing the contacts into an array have been spotted, code sending HTTP requests too, but how the contacts array is posted is yet unclear.

Fig. 3: SMS message sent by SymbOS/Yxes.A!worm - credits to hi.baidu.com. The message can roughly be translated to " [global Chinese adult online][...] free to join, valid in 3 days" + the link to a malicious web server.

spread to the memory card (i.e replicate) but that those files are created as temporary files to contain the newly downloaded malware.

6 WinCE/Redoc

Redoc is a family of malware for Windows Mobile devices, with several variants. They share similar goals (sending SMS, scheduling tasks etc) and the fact all are Portable Executables (PE) meant to run over the .NET Compact Framework platform and are written with the Basic4ppc tool.

6.1 Infection

End-users get (unintentionally) infected by Redoc by downloading and running the malware.

6.2 Payload

The exact payload of Redoc samples vary, but usually they send SMS messages, and launch customizable executables or alarms.

The analysis of Redoc can be done with a .NET decompiler such as RedGate's .NET Reflector. It reveals two major parts:

- a compiled resource, named B4Pruntime.n.b4p (or B4Pdesktop.n.b4p - depending on which Basic4ppc tool was used to develop the sample). The payload of the malware is in there.
- an interpreter, named B4Pruntime.exe. Its Main() function loads the resource (see next item), decodes it and executes the statements.

So, the payload of the malware is located in the resource. As it is compiled, it is not immediately understandable to the analyst: a reverse engineering tool must be written for this task. Having a look at the interpreter's Main() function, we find that the resource contains sections such as "Label Place" collection, "Stub Place" collection, "Line" statement collection, "Object" name and "Assembly" name list. The "Line" section is disassembled

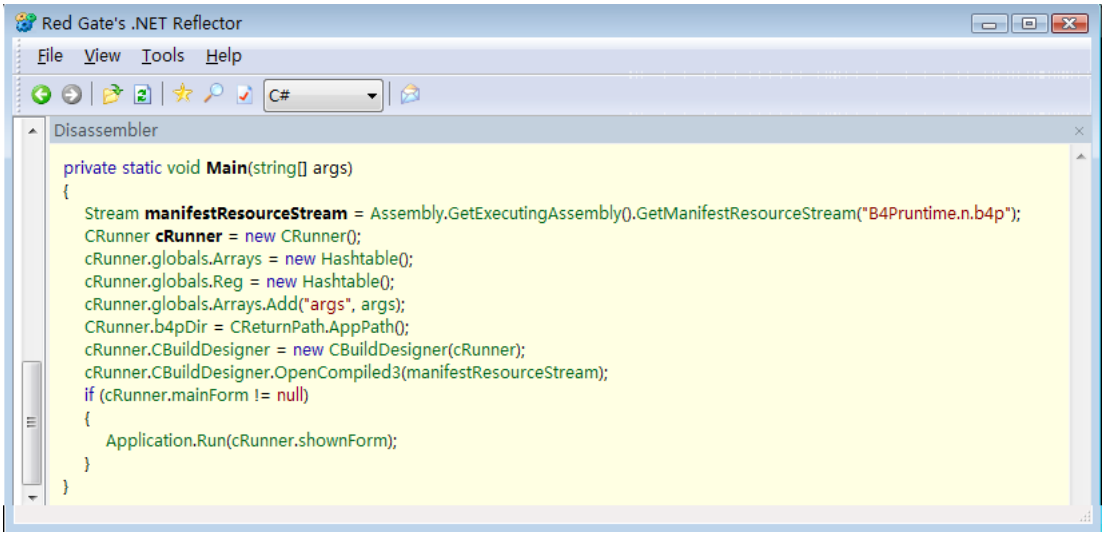

Fig. 4: Decompilation of B4P interpreter's Main() function

into readable code such as the following (this one corresponds to WinCE/Redoc.D!tr [For09c]) :

```
>>> BEGIN LINE LIST:
COUNT = 9
[1]  _main_globals
[2]  end sub
[3]  _main_app_start
[4]  _main_cnf . new1 ( 3833 , suloto )
[5]  _main_hrd . new1
[6]  _main_t = ( 03:32 )
[7]  _main_v = ( _main_t , 0 , 0 , 1 )
[8]  _main_hrd . runappattime ( _main_hrd . getspecialfolder(
                    _main_hrd . sfwindows ) & /cldll.exe , _main_v )
[9]  end sub
>>> END LINE LIST
```

This code corresponds to 2 methods (_main_globals and _main_app_start) of the Main class. The malicious code is contained in _main_app_start.

The disassembly of the Objects sections explains that the _main_cnf object is an SMSMessage object.

```
>>> OBJECT TEXT: _main_hrd:Hardware
_main_cnf:SMSMessage
```

So, we understand that an SMS message is sent to the short number 3883, with the text 'suloto' (new1 method on the _main_cnf object). Sending SMS to short numbers

are surcharged and the income usually goes to the related service (minus a short number rental fee). Additionally, an executable is scheduled at 03:32.

6.3 Propagation

Redoc does not have any propagation mechanism. Overall, it is a good example of simple and customizable malware.

7 Java/GameSat.A!tr

Java/GameSat.A!tr is a malicious Java ME midlet, i.e a Java application for mobile platforms. It runs on most mobile phones, as nearly all support Java.

7.1 Infection

The midlet poses as a modification to the famous Opera Mini mobile phone web browser. There is of course no relationship with Opera Mini apart from the splash screen (see Figure 5). End-users consequently get infected by downloading and running the trojan on their phone.

7.2 Payload

When launched, the midlet asks for the end-user's approval to send an SMS to the short number 151 (see Figure 6). Then, it displays a screen with a few options to register to various online non-free divination, gaming or dating services. Each time a service is selected, the midlet tries to send an SMS to 151 (asking for approval).

For this midlet, the difficulty of analysis does not lie in reversing its code: midlets are implemented in Java and can easily be reversed with Java decompilers such as DJ Decompiler or Java Decompiler GUI. Rather, the difficulty consists in figuring out what text is sent in the SMS and what it means. The midlet reads its settings from a file named regproperties.txt. The exact content of this file depends on the sample, but it typically contains information such as:

```
regkey,sdc,title,regdesc,imagepng,param1,param2,param3
"TRANSFERPULSA 0856xxxxxxx 20000","151","Game Gratis","Jika ingin ...
"TRANSFERPULSA 0856xxxxxxx 20000","151","Mama Lauren","Mama Lauren ...
```

Each line corresponds to a different service. The first parameter of each line is the SMS text, the second parameter is the SMS phone number (151). The other parameters are not relevant to this paper (note the last ones have been cut out to fit in page's width).

The SMS text begins with the string TRANSFERPULSA, then a phone number, and finally an integer. After some research on Internet, we found out that the Indonesian operator Indosat offers to subscribers of pre-paid phone cards the ability to transfer small amounts of money (Indonesian rupiah) from one account to another. So, if the GameSat trojan is installed on the phone of a Indosat customer, letting it send an SMS actually transfers 20,000 Rp to the account whose phone number is 0856xxxxxxx.

7.3 Propagation

.

Like Redoc, GameSat does not have any propagation mechanism. It relies on the fact it is downloadable from public web servers to spread.

Fig. 5: Splash screen of the Java/Gamesat.A!tr trojan

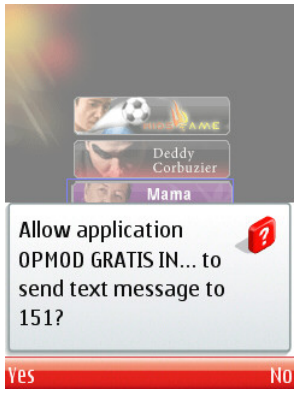

Fig. 6: Main screen of Java/GameSat.A!tr in action

7.4 Maliciousness

Readers probably now understand why GameSat is classified as a malware. It isn't detected because it is offering non-free divination services (this is not against the law) but because:

1. it is advertised as a modification of Opera Mini - which it is not.
2. it tries to transfer money from the victim's account to another account (probably controlled by a cyber-criminal) without any warning. When sending the SMS, the victim does not know what the text contains and, hence, does not know his Indosat account is about to be debited.
3. it attempts to send an SMS each time the trojan is launched, i.e even before a non-free service is selected.

8 Discussion

As detailed in Section 2, the four malware we have described in the previous sections have not been chosen at random. They have been selected for this paper because they are targeting different mobile platforms and because they are representative of most threats in 2009.

Table 1 summarizes the four malware. The platform column indicates on which platforms the malware is able to run. Only Java malware are really able to run over several different operating systems. The propagation column indicates how the malware spreads to other devices. The intent column explains what seems to be the real goal of the malware. The skills column summarizes how difficult it is for a developer to implement the malware. Finally, the vulnerabilities column shows whether the malware exploits a specific vulnerability in his malware, or not.

The technical simplicity of the four malware is striking. GameSat and Redoc are the best illustrations of simplicity, with only a few lines of source code and absolutely no hack whatsoever. Eeki's code is simple too, with more than a few lines of Bourne shell and

Name	Platform	Propagation	Intent	Skills	Vulnerabilities
Eeki	iPhone	Network	Steal ING Direct bank passwords	Unix beginner	None
Yxes	Symbian	SMS	Unclear. Sends SMS. Debugging phase for a botnet ?	Good Symbian programming	None
Redoc	WinCE	None	Make money out of calls to premium numbers	.NET beginner	None
GameSat	Java	None	Transfer funds to a pre-paid card	Very easy	None

Tab. 1: Summary of the four selected malware

basic socket programming. The fact it logs as root with the default password is a mere misconfiguration, not an OpenSSH vulnerability. The Symbian Yxes worm is the most complicated of the four, yet it only uses public APIs, no vulnerability. So, even if security researchers, used to looking into advanced issues, often forget to take basic threats into account, it is important to highlight the KISS paradigm (Keep It Simple and Stupid) works very well for malware. No need to build sophisticated threats if simple ones already provide a nice return on investment. With over 4 *billion* mobile phones in the world, finding a few victims is (usually) not a problem...

Besides simplicity, we could also note that, for the four malware, the funeral concerns the victim's bank account. Redoc and Yxes send SMS messages without user's consent: those SMS are billed to the victim unless his subscription includes free SMS. With GameSat, funds from the victim's account are silently transferred to another account. And with Eeki, malware authors try to steal banking credentials (no need to explain what they will do with that). So, in all four cases, the major impact for an infected end-user is on his bank account. Those malware do not try to lock the handset, make it unusable or any other nuisance. They only try to make money, perhaps only a few cents each time, but a substantial sum in the end. For instance, consider the business model of Redoc. In Europe, a short phone number can usually be rented for a few hundred euros per year. Each SMS sent to that number may cost up to 3 euros. Half of it goes to the management of the short number, the other half to the malware authors. This means that the business starts to be profitable over only 83 SMS ! With 20,000 SMS sent (the figure is still pessimistic for a virus which potentially propagates to millions of subscribers), the malware authors make approximately 30,000 euros.

On PCs, money has driven cyber-criminals for years [Lov06], yet, on mobile phones, people still commonly think this is a land for lonely hackers. Those four malware show this is changing and that even simple malware can be extremely profitable for their authors.

Are we beginning an era of mobile cyber-delinquents ?

9 Conclusion

This paper has described four different malware for mobile phones, among the most prevalent in 2009: a Java ME midlet, a .NET application for Windows Mobile, a Symbian and an iPhone worm. The description has shown how the malware infect the phones, their payload and how they propagate.

At least two trends have been noticed among those malware. First, their simplicity is striking and certainly raises questions whereas how mobile phones can best be secured. Second, their growing interest for money-making schemes is another cause for concern, as it means their authors are no longer disinterested hackers. Those trends will need to be confirmed in the next few years.

To block those malware, a few counter-measures are possible. Of course, firewalls and anti-virus applications would be helpful, either directly on the mobile or on the operator's network. Mobile anti-virus solutions exist but are not wide spread yet. Explicitly stating how much a given non-free operation is about to cost is another solution, but it is likely operators will be reluctant to do so and, any way, the whole process might end up to be tedious to the end user. Compartmentalizing processes, each in its security zone so that it cannot infect or access another category of processes, is another idea ARM's TrustZone processor or virtualization worlds have started to explore. They yet need to be applied to viruses and worms.

References

[Apv09a] Axelle Apvrille. Duh's not malicious dude !, December 2009. http://blog.fortinet.com/duhs-not-malicious-dude.

[Apv09b] Axelle Apvrille. SymbOS/Yxes or downloading customized content, July 2009. http://blog.fortinet.com/symbosyxes-or-downloading-customized-malware/.

[Cur94] Richard Curtis. Four Weddings and a Funeral, 1994. http://en.wikipedia.org/wiki/Four_Weddings_and_a_Funeral.

[Cyb09] Cyberinsecure. Mobile Malware Transmitter.C Spreading In The Wild, July 2009. http://cyberinsecure.com/mobile-malware-transmitterc-spreading-in-the-wild/.

[Dan09] Dancho Danchev. New Symbian-based mobile worm circulating in the wild, February 2009. http://blogs.zdnet.com/security/?p=2617.

[For09a] Java/GameSat.A!tr. Fortiguard Center, Virus Encyclopedia, 2009. http://www.fortiguard.com/encyclopedia/virus/java_gamesat.a!tr.html.

[For09b] SymbOS/Yxes.E!worm. Fortiguard Center, Virus Encyclopedia, 2009. http://www.fortiguard.com/encyclopedia/virus/symbos_yxes.e!worm.html.

[For09c] WinCE/Redoc.D!tr. Fortiguard Center, Virus Encyclopedia, 2009. http://www.fortiguard.com/encyclopedia/virus/wince_redoc.d!tr.html.

[Gos08] Alexander Gostev. Malware Evolution: January - March 2008, May 2008. http://www.viruslist.com/en/analysis?pubid=204792002#l5.

[Hyp07] Mikko Hypponen. Mobile Malware. In *16th USENIX Security Symposium*, August 2007. Invited talk.

[Lov06] Guillaume Lovet. Dirty money on the wires: the business models of cyber criminals. In *Virus Bulletin Conference*, 2006.

[McM09] Robert McMillan. First iPhone Worm Spreads Rick Astley Wallpaper, November 2009.

[MM09] Collin Mulliner and Charlie Miller. Fuzzing the Phone in your Phone. In *BlackHat USA*, June 2009.

[Oci] Solutions mobiles (in French). http://www.ocito.com/solutions-mobiles-25.html.

[PSY09] Phillip Porras, Hassen Saidi, and Vinod Yegneswaran. An Analysis of the Ikee.B (Duh) iphone Botnet. Technical report, SRI International, 333 Ravenswood Avenue, Menlo Park CA 94025 USA, December 2009. http://mtc.sri.com/iphone/.

[SN07] Shub-Nigurrath. Primer in Reversing Symbian S60 Applications, June 2007. Version 1.4.

[Zha07] Jie Zhang. Find out the 'Bad guys' on the Symbian. In *Association of Anti Virus Asia Researchers Conference*, 2007.