# Symbian Malware: An analysis of the Causes, Trends and Future.

11-July-2007


## Abstract

Over the past few years, computer scientists have been witnessing the evolution of a vicious species a.k.a. Mobile malware!! Mobile devices like Smart phones, PDAs and palmtops have become an essential part of everyday life. You take control over a person‘s cell phone and you have control over the person‘s entire life (his/her family details, bank accounts, salary details and what not?). Now, how‘s that for power!! No wonder there are so many misdirected people out there trying to infect the mobile environment with viruses which would give them unlimited access to a victim‘s personal, professional and financial life. Among mobile devices, the worst affected are Symbian smart phones. Symbian is an extremely popular OS used in a variety of smart phones. The popularity of Symbian and the consequent large pool of Symbian phone users have aroused the interest of hackers and virus writers alike. More than 90% of the current mobile malware targets Symbian OS phones. In this paper we look at some of the existing Symbian worms, viruses and Trojans and trace the vulnerabilities exploited by these malicious programs. We also point out the subtle similarities and differences between these malicious programs and then look at the future of mobile malware.



***Keywords:*** *Symbian malware, Mobile worms, Mobile viruses, Mobile Trojans, Mobile spyware.*


## 1 Introduction

The word is out. Mobile viruses are a reality. The long standing debate about whether mobile viruses really pose any threat is over. Virus writers have shown that they mean business. And going by the current trend it is clear that they have ambitious plans of expansion. Antivirus companies now have hundreds of Trojans and worms for mobile phones in their collections. What started in 2004 as a drizzle of malicious programs has now slowly turned into a downpour and by the looks of it we can expect a thunderstorm in the near future. Every week about ten mobile phone Trojans are added to antivirus databases. Symbian phones have been among the worst affected. The popularity of Symbian and the consequent large pool of Symbian phone users have aroused the interest of hackers and virus writers. More than 90% of the current mobile malware targets Symbian OS phones.

Perhaps the most disturbing threat is that of the user′s privacy. A mobile phone is like your trusted friend who knows everything about you. It acts as a single source of information for all your personal data like phone numbers, messages, agenda and much more. Imagine what would happen if someone corners this trusted friend of yours and starts extracting all information about you. Sensitive data can be deleted, modified or stolen. Chaos every-

where!! Definitely we wouldnt́ like to be at the receiving end of such a vicious attack.

Another disturbing fact is that malware writers seem to be competing with Symbian OS developers. They are continuously looking for new ways to hack into mobile phones. The more the sophisticated features introduced by Symbian the better are the ingenious ways used by malware writers to attack these phones. Gone are the days when mobile phones were simple communication devices. Present generation mobile phones are full-fledged multimedia devices with little to differentiate them from palmtops. Considering these facts, it wouldnt́ be incorrect to say that the kind of security problems of mobile users would be disturbingly similar to those already faced by PC users.

Another interesting fact worth mentioning is that while the growth rate of mobile malware has far outpaced the growth rate of its Desktop counterpart, the rate at which the awareness about mobile malware is spreading is much slower. Ask an average user to install an antivirus for his mobile phone and he will probably laugh at you. Thereś a long way to go before users know as much about mobile viruses as they do about computer viruses.

The remaining part of this paper is organized as follows: In section 2 we give an introduction to Symbian OS and point out some of its vulnerabilities which are exploited by malware writers. In section 3 we look at the three major families of Symbian malware. In section 4 we look at some other malware families and point out how they borrow their traits from their predecessors. In section 5 we look at the steps taken by Symbian in response to this malware threat. In section 6 we look at the future of mobile malware. Finally we end with our concluding remarks in section 7.

# 2 Symbian OS

Having a good knowledge of the basic structure of Symbian would be useful in the analysis of Symbian malware. However, it would be impossible to discuss the entire architecture and all the features provided by Symbian OS in a single paper (even an entire book would not be sufficient for this). Here our aim is just to introduce the reader to the basic structure of Symbian OS and some of the features provided by it. We will look at each of the following topics in some detail:

1. Symbian file system.
2. The APIs provided by Symbian for writing applications.
3. SIS file format.

As we go along we will also point out some vulnerabilities/features that are exploited by malware.

## 2.1 Symbian file system

Just like desktop PCs Symbian file system is based on drive letters and directories. A typical smartphine has the following directories:

1. **C-drive:** A non-volatile, writable storage area (phone memory). (This is the place where the viruses get installed)
2. **D-drive:** A volatile temporary storage area reserved from RAM
3. **E-drive:** A non-volatile removable, writable (usually) memory card (which come in different types). (Can be used for spreading viruses from one device to another)
4. **Z-drive:** A non-volatile, non-writable storage area (where the firmware/operating system resides)

**Vulnerabilty 1:**

**Overwriting ROM Applications**

If an application on C: has the same name and path as one in Z: then it will get executed instead of the original application in the Z: drive.

E.g. C:\ system\ apps\ phonebook.app will override Z:\ system\ apps\ phonebook.app

As we will see later, this vulnerabilty is exploited by a number of Trojans to disable system applications and/or override them with malicious applications

But, why did Symbian provide such a dangerous feature. Was it a bug? Well, we wouldnt́ say that it was a bug. This feature was deliberately included to allow patching of a binary in ROM without needing to re-flash the device. Alas!! The virus writers were smart enough to exploit this feature as a source of personal entertainment!!

## 2.2 Symbian APIs

Symbian provides a wide range of C++ APIs for application development. An exhaustive set of APIs is available for:

1. GUI development.
2. SMS, MMS application development.
3. Bluetooth application development.
4. File system access.
5. Accessing phonebook and other items in phone memory.

This complete set of APIs would be any application developerś paradise. Imagine having full access to phone memory and other features like SMS, MMS and Bluetooth.

**Vulnerabilty/Feature 2:**

**Extensive Set of APIs**

*Note: Itś sad that we have to call this a weakness instead of calling it a feature.*

This extensive API set is one of the main reasons for Symbianś popularity. It is also the reason for the exponential growth of malware for Symbian OS as it opens up a lot of possibilities for writing malicious code. This is what goes on in a virus writerś mind:

1. I could use the Bluetooth APIs to scan for devices in my neighborhood (within a range of 10 m) and then send some malicious file using an obex client. Symbian provides classes like RSocketServ, TProtocolDesc, RHostResolver and CObexClient which can be used for this. In fact, the Symbian site also has tutorials on how to use these APIs.
2. I could access the phonebook, read all contact information and send SMS to all the contacts and cause financial loss to the victim. Symbian provides classes like CContactDatabase(for accessing phonebook) and CsmsClientMtm (for SMS) which can be used for this.
3. I could access the phonebook, read all contact information and send a malicious file as an attachment using MMS. CMmsClientMtm can be used for this.

The point to be noted is that much of the work of a virus writer is simplified by these APIs. He does not have to worry about using hacking techniques to get access to MMS, phonebook etc. The APIs provide him/her these facilities which makes it somewhat easy to develop viruses for this OS.

## 2.3 SIS file format

Software Installation System (SIS) files are the only currently known method for a normal user to import executable code to a Symbian device. Any malware that wants

to run on the device has to get installed as a SIS file. Thus all known malware uses SIS files. In the earlier versions of Symbian OS (upto v8.1) no security features were included to prevent the tampering of SIS files which made them vulnerable. To add to this, the format of these SIS files was openly known and available on the internet. These facts make way for the following vulnerabilties:

**Vulnerabilty 3:**

**Easy replication of SIS files**

Malware writers can write malware that could replicate by constructing its own SIS file programmatically i.e. embed code in the malware to construct its own SIS file. This requires a deep understanding of the structure of the SIS file and the offset of various fields in it. But as we said earlier these details are easily available on the internet and any malware writer worth his salt should be able to do this easily. (Readers familiar with constructing TCP/IP or MPEG-2 packets would know that this is not a difficult task and simply involves bit/byte/word manipulations.)

**Vulnerabilty 4:**

**Tampering existing SIS files**

Malware writers can write malware that searches for existing trusted SIS files on the target device and then embed the SIS file of the malware into these trusted SIS files so that the malware can propagate along with the trusted SIS file. This is possible because Symbian allows a SIS file to have embedded SIS files. Furthermore, it also provides an option wherein the embedded SIS file gets installed automatically along with the original SIS file. Again the above procedure requires a deep understanding of the structure of the SIS file and the offset of various fields in it.

## 2.4 Symbian Executables/Resource Files and Font files

The following files are typically available on a Symbian smartphone:

1. **Foo.app:** These are the applications which are visible to end users and are accessible from application menu.

2. **Foo.exe:** These cannot be accessed by the end user. These are services or utilities that are used by GUI applications

3. **Foo.mdl:** They have the ability to start automatically at boot or when a memory card is inserted. These files must be located in the System\Recogs directory of one of the drives.

4. **Foo.rsc:** These are resource files for the application

5. **Foo.gdr:** These are Symbian OS font files.

**Vulnerabilty 5:**

**Autostart Capability**

*Note: Again a good feature which was used maliciously by malware writers.*

Once a malware is installed on a victim device it would be desirable that it gets executed automatically without any user intervention. This feature is provided by Symbian in the form of .mdl files which can be configured and programmed to start any application (including a .exe or a .app file). This feature was exploited by virus writers by packaging .mdl files along with their malicious .app/.exe files. The purpose of these .mdl files was to start the malicious code whenever the system reboots (without requiring any user intervention).

**Vulnerabilty 6:**

**Malformed files**

It has been observed that some of the older versions of Symbian OS (upto v8.1) had a design/implementation flaw which caused the device to hang/reboot whenever it encountered a malformed .rsc/.dgr/.exe file. Malware writers were quick to discover this and exploited this vulnerability by packaging malformed files along with their applications. Most malware writers exploit these vulnerability in combination with the Autostart Capability feature by including a .mdl file in the SIS file. The purpose of this .mdl file is to try to execute these malformed files when the system boots or to execute some dummy application which uses these malformed files. Since these files are malformed and do not adhere to their standard format the system fails to execute/process them and freezes.

# 3 The Most Wanted

Having seen the vulnerabilities present in Symbian OS we will now start tracing the evolution cycle of Symbian malware. Itś a known fact that most of the new worms and viruses borrow heavily from their predecessors. As and when the source codes/technical details of existing worms become available, more and more script kiddies capitalize on the work of other malware writers and come up with new variants of these worms. It is therefore necessary to first look at the pioneers (for the want of a better word) in the field of worms and viruses and then draw comparisons of newer worms with these path breakers (or should we say security breakers). Hence we begin with a brief description of the 3 most path breaking malicious programs and then move on to discuss about other malware which borrow their malicious intentions and techniques from these malicious programs.

## 3.1 Cabir

Cabir was written by a group of virus writers called 29A. Their intention was to write a proof-of-concept malware for mobile phones and other devices running non-standard operating systems and applications. The authors definitely succeeded in their aim because this was the first time that virus researchers encountered a worm which:

1. was written for a non-standard operating system (Symbian)

2. was meant to run on a different processor (i.e. ARM till the time when Cabir was released researchers were familiar only with worms for the x86 processor which is commonly used in Desktop PCs.)

3. used a different medium for propagation. (Bluetooth - as opposed to e-mail/ internet)

Cabir exploits the rich set fo APIs provided by Symbian for achieving its malicious ends. The writers used Symbianś Bluetooth APIs for discovering vulnerable devices and infecting them. They also did a deep study of the SIS file format and were successful in writing code which had the ability to replicate itself (i.e. code which could construct its own SIS file programatically). The SIS file also included a .mdl file which launches the worm automatically on system reboot.

## 3.2 CommWarrior

CommWarrior takes the credit for being the first known mobile phone virus capable of spreading via MMS messages and thereby causing financial losses to the user. There are many similarities between Cabir and CommWarrior as listed below:

1. Both the worms run in the background and continuously search for and spread to potential victims.

2. Both the worms are capable of spreading via Bluetooth (In addition CommWarrior is also capable of spreading via MMS which makes it much more lethal as compared to Cabir as MMS has no boundaries and can be instantly sent even to handsets in other countries.). Thus both exploit the rich set of APIs provided by Symbian.

3. Both the worms use the same strategy for replication (i.e. programmatically create their own SIS files).

### 3.3 Skuller

Until the arrival of this Trojan, virus writing was an art (evil as it maybe, an art is an art is an art) which required a bit of ingenuity on the part of the virus writers. But Skuller broke the myth that virus writing was a work of nerds who had a deep understanding of the Symbian system. This Trojan showed that any person who can use a utility for creating sis files will be able to create a Trojan of this kind. The rest of the work is done by the vulnerabilities (vulnerability 1 and 6 to be precise) present in Symbian. It is possible to overwrite any files, including system files, and the system becomes very unstable when it comes across unexpected files. Moreover, being a Trojan, it does not require any ingenious ways of propagation/replication. It relies on the curiosity of over eager users to download anything and everything with a catchy caption like Extended Theme for your Symbian Phone and the users oblige!! The earliest evidences of this Trojan date back to November 2004. It overwrites all the default applications that are installed on any Symbian OS phone (refer vulnerability 1 above). The application files created by Skuller are standard application files for the Symbian platform and do not contain any malicious code (it is simply dummy code which doesnt́ do anything). This Trojan also replaces the icons of all the applications with images of skulls.

## 4 The Followers

*If you did it, so can I!! Better (Deadlier) than thy!!*

An insatiable hunger to prove their superiority attracts most of the virus writers to this evil field. Each time a malware writer comes up with a new malicious program, there are a 100 others eagerly burning the midnight oil to out perform him/her. This is perhaps the reason that since the Pandoraś box was opened by Cabir there has never been a shortage in the production of malicious programs. One important observation worth mentioning is that as time passes the process of malware writing becomes somewhat similar to the process used by a chef to come up with newer tastier dishes:

> *Take a few existing recipes and add or modify a few ingredients and there you are!! You have a new dish!! More delicious than ever before!!*

(Is it a coincidence that delicious and malicious rhyme??)

Keeping this phenomenon in mind, we now look at some of the malware that followed in the footsteps of the famous three. Of course, some of these worms have their own identity and dont́ directly borrow from their predecessors. But, as we shall see, the basic idea still remains the same:

1. Use the connectivity (Bluetooth, MMS, SMS) APIs provided by Symbian for performing malicious activities.

2. Replace system binaries (Overwriting ROM applications).

3. Replace system configuration files with corrupted/malformed files.

The table in Appendix A explains the vulnerabilities that these malicious programs exploit and also compares them to their predecessors.

# 5 Symbian rises to the challenge

*Nero fiddled while Rome burned. Or did he??*

Well, one might be tempted to ask that While Symbian phones were being attacked at all fronts, what were the OS developers doing?? Were they sleeping or secretly enjoying the rampage?. Well definitely not. Symbian OS developers pulled up their socks and some serious decisions were taken to enhance Symbianś security model. And indeed, Symbianś security model has evolved over time and reached a stage where it can put up a strong defense against mobile malware. Below, we summarize the evolution process of Symbianś security model.

1. **Symbian OS v7 and earlier:** No security checks present. User decides to install based on dialog prompts.
2. **Symbian OS v8:** Anti-virus support was introduced.
3. **Symbian Signed:** Developer needs to take a digital certificate from Symbian.
4. **Symbian OS v9:** Runtime security checks were performed and the concept of capabilities was introduced. Data caging and UID caging were introduced to malicious or unintentional overwriting of one applicationś data by another.

Now we briefly look at the key security features introduced in Symbian OS v9.0 which make it resistant to most of the worms, viruses and Trojans that we discussed in the earlier sections.

## 5.1 Runtime security check and capabilities

In Symbian OS v9 there is a more fine-grained security model which allows privileged access to be granted based on capabilities. These capabilities are based on clearly defined groupings of what each API is designed to do. The unit of trust is at the process level so a process is only able to access resources for which it has the relevant privilege. The process cannot use APIs or resources that require more capabilities than have been authorized. This security check for each process is policed at runtime.

**Example:** If an API is associated with network services (Bluetooth, SMS and MMS) it will require the capability NetworkServices. This implies that if a malicious program uses the above mentioned network APIs for propagation or simply for causing financial damages to the victimś device then the writer of such malicious programs needs to get the corresponding permission from Symbian in the form of a digital certificate (which implies that he/she can be easily traced). ***So goodbye to Cabir, CommWarrior and similar viruses, worms and Trojans which use network services (or any other APIs which require some capabilities) for performing malicious activities!!***

*Note: There is a small twist to this happy tale. If a program tries to use an API for which it does not have the necessary privileges then the user will be alerted about this. The user then has an option to bypass the security check and allow the access. Uh-oh, sounds like trouble!! We have already seen how over enthusiastic users will be more than happy to ignore such security warnings. An inherent flaw here is that the system partly depends on the user*

*to ensure security.*

### 5.2 UID usage and data caging

To prevent malicious or unintentional overwriting of one applicationś data by another, Symbian OS v9 introduces a change in the way UIDs are managed that allows for data caging. This ensures that most applications can only access areas of the system designed as public or private to that particular application i.e. neither read nor overwrite the data belonging to other applications in order that data associated with each particular application remains secure. ***So goodbye to Skullers and similar Trojans which overwrite the data of system and third party applications!!***

*Note: Again, there is a small twist to this happy tale. If a program tries to overwrite the files of some other application then the user will be alerted about it. The user then has an option to bypass the security check and allow the access. Uh-oh, sounds like trouble!! We have already seen how over enthusiastic users will be more than happy to ignore such security warnings. An inherent flaw here is that the system partly depends on the user to ensure security.*

## 6 The future of mobile malware

In this section we talk about some possibilities which might become realities in the not-so-distant future.

### 6.1 The next big (dirty) thing: Mobile spyware

We have not really looked at mobile spyware in detail and have only made passing references to it. Mobile spyware would be a logical extension of mobile malware. We have mentioned some Trojans (PbStealer) which behave as spyware and steal sensitive data from the users mobile. But what if this ***Spywar*** (not a spelling mistake) is taken to the next level? Would it be possible to create spyware which uses the Camera APIs of your phones to click pictures and send them to the attacker using MMS!! (My!! My!! Now that would be complete breach of privacy!!) As far as our knowledge goes it should be possible to this. We would like to explore this possibility and alert the Symbian authorities in time (before anyone else releases such a spyware in the wild).

### 6.2 Polymorphic Mobile Worms

Again, this is a possibility which has never been explored before. There are hordes of polymorphic worms available for Desktop PCs. In fact there are several polymorphic engines which are openly available for download. We wouldnt́ be wrong in saying that sooner or later this would happen in the mobile arena also. Itś only the question of when an overly gifted and overly enthusiastic virus writer develops a polymorphic engine (for the ARM processor) and makes it available to the brotherhood of virus writers.

### 6.3 Cryptovirolgy

Cryptovirology refers to the use of well established cryptographic algorithms for performing malicious deeds. Cryptovirology is already a menace in the Desktop arena. Its only a matter of time before it reaches the mobile arena. As an example consider this. Suppose a malicious user installs a Trojan which encrypts all the data on your phone using the malicious user‘s public key. Now the malicious user can hold you at ransom and demand some money in exchange for releasing his private key (so that you can decrypt your data). This has never been tried before but seems possible.

# 7 Conclusion

*And they lived happily ever after!! Or did they??*

We wish we could end our report on a happy note saying that *Alls well! Go Home!! Relax!! Your privacy is guaranteed!!* But unfortunately, you know, as much as we do, that this is not true. We have only looked at the tip of the iceberg and from what we have seen it is pretty clear that things dont́ look very safe. What makes matters worst is the Chef Recipe Theory that we mentioned in our report. A little bit of salt, a little bit of pepper and you have a new dish. This theory definitely explains the steady flow of mobile malware and gives us every reason to believe that this steady current will only increase in the future till it becomes strong enough to wipe out the entire community of mobile users. Now, hold on, thatś definitely an exaggeration and is based on the assumption that Nero will continue fiddling while Rome burns!! Fortunately that is not the case here. Symbian has definitely risen to the challenge and done its best to get a hold over the situation. The Symbian Signed initiative seems to be a step in the right direction but even this system is not foolproof because of its partial dependence on sensible behavior by the users. Another important point to understand is that Symbian Signed works only with Symbian OS v9.x (and we hope it would work with the later versions as and when they are introduced by Symbian). But there are already a large number of users using Symbian v7.x and v8.x phones which are susceptible to many malicious programs (and the number of such programs will only increase in the future). These users are living under a constant threat of being attacked by mobile malware. There is no immediate remedy to protect these users. Itś true that several anti-virus companies have developed and distributed free anti-virus for mobile phones but then a question which remains to be answered is that *Is an average mobile user (security) conscious enough to realize the importance of installing and maintaining such anti-virus programs on his device?.* No, definitely not!! What anti-virus companies provide is cure and what we are looking for is precaution.

The only possible way of completely eliminating the threat posed by mobile malware is to ensure that no malicious program ever reaches the phone of an unsuspecting user. Once it reaches the victimś device there is nothing we can do as in all probability the victim will happily install it on his phone. One possible way of preventing malware from reaching a victimś device is if the service provider does content based filtering of all MMS messages being sent over the network. This will take care of the MMS based worms and viruses but then what about worms which spread over Bluetooth? Also it is important to note that the world of mobile malware is dominated by Trojans and not by worms or viruses. These programs do not need any propagation vector and simply rely on the userś curiosity to download and install them. Trojans pose a much greater threat than worms and viruses because they are relatively much easier to write and in most of the cases do not require any ingenuity on the part of the writer. Even with the Symbian Signed initiative it would be very difficult to get completely rid of such Trojan programs. They will continue to reign and cause havoc for at least the next few years.

*Never underestimate thy enemies!!*

Malware writers have known to be able to find ways to hack into a system, no matter how secure it is. Secure as it may look (and be), it would be foolish to assume that malware writers would never be able to hack into Symbian Signed. Only time will say what this gifted but

misdirected community of virus writers has in store for mobile users. Till then,[1]

***Stay Alert!! Stay Secured!!***

## References


[1] ***Symbian OS***, *www.symbian.com.*

[2] ***Symbian C++ APIs***, *http://www.symbian.com/developer/techlib/v70docs/$sdl_v$7.0/doc_source/reference/cpp/index.html.*

[3] ***Virus Bulletin. VB2005***, *https://www.virusbtn.com/pdf/conference_slides/2005/JNiemela_VB2005_sanitized.pdf.*

[4] ***Thouky. mtlworld.*** *http://homepage.ntlworld.com/thouky/software/psifs/sis.html.*

[5] ***Symbian SIS files v9.1.*** *http://developer.symbian.com/main/downloads/papers/SymbianOSv91/softwareinstallsis.pdf.*

[6] ***Symbian Signed*** *https://www.symbiansigned.com.*

[7] ***F-Secure*** *http://www.f-secure.com/.*

[8] ***Mobile Evolution. Viruslist.*** *http://www.viruslist.com/en/analysis?pubid=200119916beg..*

[9] ***Virus List*** *http://www.viruslist.com.*

[10] ***Yahoo Groups.*** *http://tech.groups.yahoo.com/group/SymbWarrior/.*

[11] ***Nokia Discussion Forum.*** *http://discussion.forum.nokia.com/.*

[12] ***Symbian Security evolution.*** *https://www.symbiansigned.com/How_has_Symbian_Signed_evolved_with_Symbian_OS_v9.pdf.*

[13] Fadia, Ankit. ***The Ethical Hacking Guide to Hacking Mobile Phones.*** Macmillan India Ltd. (India), Thomson Learning (International), 2006.

[14] Harrison, Richard. ***Symbian OS C++ for Mobile Phones.*** John Wiley and Sons Ltd, 2003.

Appendix

# A List of Symbian Malware and the vulnerabilities which they exploit

| Sr. No. | Family Name | Type | Vulnerabilities that it exploits | Borrows its traits from | Remarks |
|---|---|---|---|---|---|
| 1 | Cabir (June 2004) | Worm | 2, 3 and 5. Uses Bluetooth and File APIs. Includes code to generate its own SIS file | Original Worm | First worm which showed the world how to replicate. |
| 2 | CommWarrior (March 2005) | Worm | 2,3 and 5. Uses Bluetooth MMS and File APIs. Includes code to generate its own SIS file | Original Worm | First worm to cause financial damages via MMS |
| 3 | Skullers (November 2004) | Trojan | 1. Overwrites all System applications with dummy applications. | Original Trojan | First Trojan to exploit an inbuilt vulnerability in Symbian OS. Showed the world how easy it is to write Trojans. |
| 4 | Mosquit (August 2004) | Trojan | 2. Uses SMS APIs | Original Trojan | Sends SMS to premium numbers. |
| 5 | Lasco (Jan 2005) | Worm | 2,3,4 and 5 | Cabir | Also tampers existing SIS files by embedding its own SIS file with them. |
| 6 | Locknut (Feb 2005) | Trojan | 6 | | Contains malformed .app and .rsc files |
| 7 | Dampig (March 2005) | Trojan | 1 | Skullers | Disables some critical system applications |
| 8 | Drever (March 2005) | Trojan | 2. Uses File APIs to overwrite the boot loaders of anti-viruses. | | Disables automatic startup of antivirus programs. |
| 9 | Mabir (April 2005) | Worm | 2,3 and 5 | Cabir and CommWarrior | Spreads via Bluetooth/MMS |
| 10 | Fontal (April 2005) | Trojan | 6 | Locknut | Installs corrupted font files. |
| 11 | Hobbes (April 2005) | Trojan | 1 and 6. Installs corrupted binaries | Skullers | Causes application loader to crash in older versions of Symbian. |
| 12 | Appdisabler (May 2005) | Trojan | 1 | Skullers | Disables critical system applications. Also installs variants of Cabir, Locknut and Skulls |
| 13 | Doomboot (May 2005) | Trojan | 1 and 6 | Skullers | Replaces system applications with malformed binaries. |
| 14 | Blankfont (August 2005) | Trojan | 6 | Fontal | Installs corrupted font files. |
| 15 | Skudoo (August 2005) | Trojan | 1,2,3,5 and 6 | Mix of some worms and Trojans | Installs some variants of Skullers, Cabir and Doomboot. |
| 16 | SingleJump (August 2005) | Trojan | 1 | Skullers | Also installs some variants of Fontal, Blackfont, Appdisabler etc. |
| 17 | Cardtrap (Sep 2005) | Trojan | 1. Disables several critical system applications apart from installing 3 Windows worms. | Skullers | The first crossover worm. It installs three Windows worms onto the devicés memory card. It also installs an autorun.ini file so that if the card is inserted into a PC using Windows then the autorun file will try to execute these worms. |

| | | | | | |
|---|---|---|---|---|---|
| 18 | Cardblock (September 2005) | Trojan | 2. Uses some APIs (DMMCController) provided by Symbian to set a random password for the MMC | | Blocks the MMC card inserted into the phone by generating a random password and setting this password to the MMC card |
| 19 | PbStealer (Nov 2005) | Trojan | 2. Uses Bluetooth, File and ContactDatabase APIs | | Dumps the contents of the contacts database into a text file and sends it to the first Bluetooth device. |
| 20 | Bootton (Dec 2005) | Trojan | 1 and 2. Uses SysStartup APIs provided by Symbian | Skullers | Overwrites system applications with a malicious program that causes the device to reboot. |
| 21 | StealWar (March 2006) | Trojan | Same as the worms/Trojans that it drops on the device | | Basically a Worm/ Trojan dropper which install some variants of Skullers/CommWarrior/PbStealer etc. |
| 22 | Flexispy (April 2006) | Trojan | 2 | Original Spyware | Records details of voice call, SMS and sends the details to a remote server. |
| 23 | RommWar (April 2006) | Trojan | 6 | Locknut | Installs malformed system components which cause the device to freeze. |
| 24 | RomRide (April 2006) | Trojan | 1 and 6 | Skullers/ Locknut | Installs malformed system components which cause the device to freeze. |

*TODO: Add the description of the worms and Trojans discovered in the period of April 2006 to May 2007.*