INTERNATIONAL BACCALAUREATE: EXTENDED ESSAY

Computer Science HL: What is the best method of detection for Polymorphic viruses?

By Luke Geeson

VIK3

2012-13

Commented [M1]: Title – "for detection of?

Abstract

The title of my essay is: What is the best method of detection for polymorphic viruses? In this essay I will be assessing some of the commonly used virus detection methods in use today. I will be discussing the threat of viruses in general and more specifically, the threat that is the polymorphic virus. I will also be evaluating these virus detection systems in terms of their capabilities, practicalities and potential in the detection of polymorphic viruses. The scope of the essay is confined to evaluation of the main types of virus detection in use by software security services today; these include smart scanners, integrity checkers, behavior blockers, sand boxing, algorithmic scanners and code emulation. Upon examination of these main types of virus detection I have concluded that the best method of detection for polymorphic viruses is the algorithmic scanning method with its variations such as smart scanners. I have concluded this because this method can be designed for the exact purpose of locating complex polymorphic viruses and as technology develops, code emulators will not be suitable for the task of detecting polymorphic viruses. Furthermore, I conclude that generic detection methods such as integrity checking or behavior blocking are too simple when considerations are made for the complex threat that polymorphic viruses represent. Unresolved questions include the direction of virus development and technology in the future although it is certain that they will become more complex, thus reinforcing the need for complex detection systems such as the specific algorithmic scanning methods to match.

Commented [M2]: Abstract – "behavior" is American spelling. English spelling is behaviour.

Table of Contents

1.0.0 Introduction

In the modern world, computers play a very important role of society. This is why it is vital that we understand how to use computer systems. We are becoming ever more reliant on the use of the computer in order to go about our daily lives: examples include the use of email and instant messenger to communicate with friends and family across the globe which, 100 years ago, would have taken at least a week (via letter). Furthermore, it is vital that we are aware of the dangers that face us when using computers; such as viruses. If our computer systems were to stop working, it would cause large scale detrimental effects that can make our society to grind to a halt and possibly endanger life. Computer Viruses do this. Recent examples of this include: the dangers of the Alureon/DNS changer bot virus which is believed to be widespread, so much so it was published that: *'it is estimated there are still hundreds of thousands of people who just don't know they are infected.'* (1) By understanding viruses, we can find patterns in their activity in order to intercept, capture and remove them from our system before their effects are made irreversible.

Due to the nature and development of technology (and with it software), the quantity of viruses present today is vast. As a result, it would require many books to explain them, books of which would mostly likely be out-dated very soon because of the rate at which viruses are made. This graph illustrates the capabilities of a single virus:

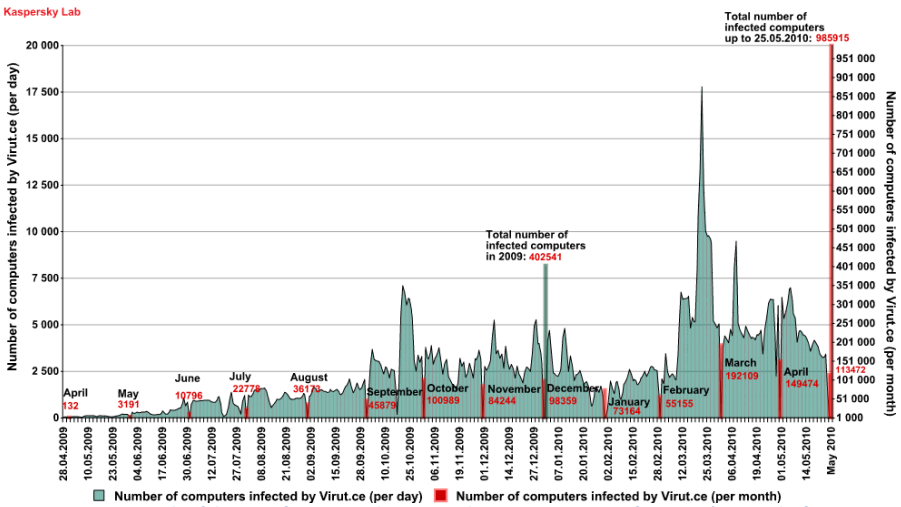

Figure 1: an example of the rate of virus growth: a Kaspersky report on activity of Virut.ce: 'Virut is the fastest-mutating virus known, with a new variant appearing as often as once a week' (2)

I will focus on one type of virus: the Polymorphic Virus. The polymorphic virus is a stealth virus that incorporates the use of polymorphism in order to avoid detection in our computer systems. The severity of this threat was realised after the 'tequila' polymorphic virus of 1991 (3) and the recent 'Flame' virus of 2012 that was 'arguably, the most complex malware ever found' (3). In addition, a polymorphic virus can incorporate the use of mutation engines to hide its signature. Thus the apparent danger is highlighted and I feel it is an important aspect of computer science that is necessary to explore in order to learn more about how to tackle these threats.

Commented [M3]: In?

Commented [M4]: Redundant.

Commented [M5]: of such uses

Commented [M6]: comma

Commented [M7]: might

Commented [M8]: -del

Commented [M9]: It,

Commented [M10]: --del

In, this essay, I will be discussing what a virus is, the polymorphic threat and the methods associated with the detection of polymorphic viruses. Furthermore, I will evaluate the methods of detection in order to judge which method presents the greatest prevention against these threats.

Commented [M11]: This is a great summary of what's coming up…good.

1.1.0 What is a computer Virus?

The British computer Society Glossary of definitions defines a virus as a 'program designed to make a computer system unreliable and to copy itself between and within computers.' (4) This is accomplished through methods such as code injection which involves the attachment of malicious files and software to files in the system and replication to recreate a file from memory. But a virus does not necessarily have *to 'destroy or illegally copy data'* (4) to be defined as a virus. Much like the biological virus, these electronic pathogens simply have to *infect* the users system without their knowledge or permission to be classed as a virus. For example, an early boot-sector virus program was the *'Brain'* virus (1986) (5) which simply renamed the drive label as *'©Brain.'*

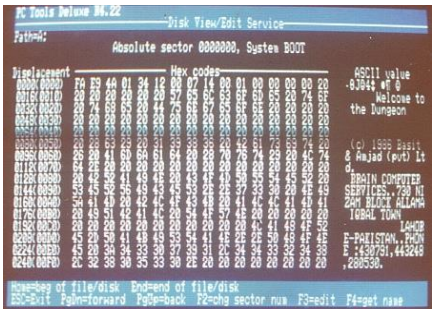

Figure 2: The hex dump boot sector of a floppy disk containing the 'brain' virus (14)

In the modern world, there are whole arrays of viruses constantly seeking to disrupt the normal operations of businesses. On British Telecom servers, approximately 120,000 viruses were detected whilst attempting to breach their systems (in July 2012). In response to the growing sophistication of cyber threats, many organisations offer services to commercial businesses to check and monitor their networks. Examples include British Telecom's Assure product range, the Information Assurance arm of GCHQ and the Vulnerability service offered by CESG (Communications-Electronics Security Group) as well as many commonplace AV companies such as Kaspersky, AVG and Symantec.

It is important to note that computers are rapidly improving: Ray Kurzweil predicts that we will 'go to the Sixth Paradigm which is 3-D molecular computing' by 2019 (6) and it is certain that virus developers will follow suit. With this, it is certain that the most complex of viruses will also advance: the polymorphic virus. These viruses have the potential to infect many systems quickly whilst remaining illusive through the use of polymorphism. Therefore it is vital that we evaluate the types of virus detection that are most prevalent in today's technology so that an assessment can be made on the most effective method finding modern complex viruses.

1.2.0 In depth analysis of Polymorphism and polymorphic viruses

In the cat and mouse game between virus developers and security firms, encrypted viruses started to appear. These comprise of two elements - a decryption routine and the encrypted virus. The encryption key is used to encrypt the virus content which changes with each new infection. This gives the virus a different digital signature with each new encryption, preventing the anti-virus software from identifying a pattern. When the infected file is launched(executed) by a user, the decryption routine takes control of the computer to decrypt the virus content and then passes control of the computer to the decrypted virus. As with simple viruses, the encrypted virus replicates and attaches itself to other files or programs. To counter this, the antivirus agencies have focussed on scanning for the bit pattern of the decryption routine itself because this remains constant. Virus developers often scramble the decryption key so that this it is harder to detect.

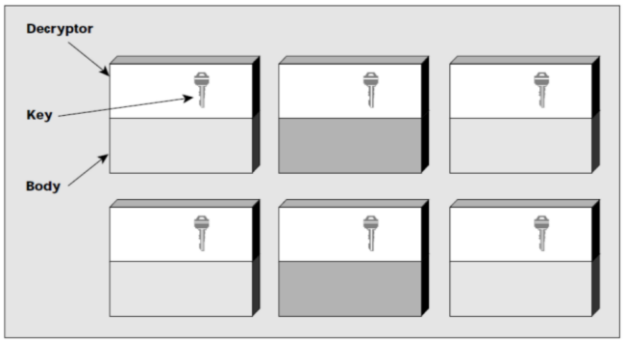

Figure 3: 'An encrypting virus always propagates using the same decryption routine however, the key value within the decryption routine changes from infection to infection. Consequently, the encrypted body of the virus also varies, depending on the key value.' (7) (The image is taken from same source)

Polymorphic viruses are particularly difficult to detect as they add a third dimension to the complexity of encrypted viruses. A mutation engine randomly changes the decryption routine with each new infection. The mutation engine is encrypted with the virus content. Once activated, the decryption routine decrypts the virus content and the mutation engine. A copy of the virus and the mutation engine are stored in memory. The virus infects a file and triggers the mutation engine to generate a random new decryption routine with no common attributes of its previous iterations.

Then, the virus encrypts a new mutation engine and the virus body attaches it to a new program. By randomly changing their digital signature, standard antivirus software cannot match the virus to any known threat patterns. Polymorphic viruses often require the costly and time consuming development of bespoke detection routines in order to tackle the threat. The sharing of mutation engines across different viruses creates yet more problems with one polymorphic being identified as another.

As a result the virus decryption routine and mutation engine will bear little or no resemblance to its predecessor with each new iteration. Security software has a lot of trouble against this formidable adversary as a result. For more information on commonplace viruses and how a typical virus will function, please refer to Appendix B: Virus definitions.

2.0.0 Development

2.1.0 How to tackle polymorphic viruses

There is a wide range of methods, functions and commands that can check for the presence of viruses. These include scanners, simulators/emulators, pattern detectors and consistency checkers. In this section I will evaluate some of the main methods of virus detection in widespread use today.

Commented [MG12]: are

Note: most generic virus scanners will use simple String sequence scanning and heuristic analysis to detect viruses and most virus detection systems will incorporate some of these features into the methods mentioned. Thus they will not have a section as they do not present unique methods of detection.

For information on technical terminology used in this section: refer to Appendix A: Relevant technical terminology

2.1.1 Integrity checking

Integrity checking will check for consistency in the system. When scanning, it will compare the checksum of a file with a trusted version stored elsewhere. If it does not match, then an alarm is raised.

Advantages

- If successful the checker can flag viruses in the early stages of infection which could potentially lead to the removal of the virus before serious damage is done.

Disadvantages

- The trusted source from which comparisons are made requires a 'clean initial state' (8); if a system already has viruses, then the check will be ineffective.
- Many applications change their own code (such as macros) as a result of updates and changes. This is bad because the integrity check will detect too many false positives.
- Certain programs use compressors to reduce the file size and with it, the check sum. This will flag a false positive if the scanner does not recognise the change.
- Integrity checking will operate when there are changes in the system. Memory resident viruses can bypass this.
- Integrity checkers are generally slow, thus some are optimised to check likely areas of infection in a file. Random overwriting viruses and entry point obscuring viruses can change files unexpectedly. Unfortunately, polymorphic viruses will randomly overwrite/infect files and so this method is quite ineffective.

2.1.2 Behaviour Blocking

As the name suggests, this is a function that can block suspect programs based on their behaviour. If a file opens an executable in order to write, the system may ask the user for permission. Unfortunately this method has problems also.

Advantages

- Is effective against large families of viruses- heuristics can be used to reduce false positives and successful infections.

Disadvantages

- Every system is different and so the 'number of behavioural patterns that can cause infections is infinite' (8) so that this scanning method cannot cater for all combinations.
- The system may become unbootable after disinfection (reflective of the above).
- The virus may have access rights (or may be memory resident) which would in turn enable it to bypass this function; polymorphic viruses may possess a digital signature which allows them to do so.
- Non-resident slow infectors may wait forever until access is granted; and so the threat will remain.
- The system may request permission too many times; this will irritate the user and pressure them to refuse to use this system.

2.1.3 Algorithmic Scanning

When generic scanners fail to detect viruses, 'virus specific' (8) algorithms are created to detect and remove the threat. String sequence scans are executed where the scanning language will seek to scan a particular area by calculating the position from the entry point or the end of a file. The OS will sequentially execute virus detection methods from a database to detect problems. Filtering is used so that scanners can search in expected areas of infection (memory resident viruses will be in the memory). C*rypto-graphic detection* technique (8) is used to attack encrypted viruses to determine encryption/decryption rules. Companies such as Norton and AVG rely heavily on Algorithmic methods as they are cost-effective when compared to generic scans.

Advantages

- Scans one object with many detection methods- scanning is effective.
- Filtering is largely effective as it detects many generic viruses based on location- it is also cost effective for companies.
- A combination of filter and static decryptor detection is a quick and effective process of scanning for decrypted viruses in suspected areas.
- There is some limitation if encrypted viruses have larger signatures than these specific scanners allow. Static decryptor detection can be used to combat this, but it is quite slow.
- Crypto-graphic detection fairs well against encrypted viruses, especially complex polymorphic viruses.

Disadvantages

- Specific algorithms prove difficult to port as they will be system specific (unless p-code is used).
- Algorithmic code can often cause system issues as updates tend to be rushed out without testing.
- Algorithmic code tends to execute rather slowly due to the precise nature of the code.

Commented [MG13]: Related to your final point, doesn't this method of scanning every file with many algorithmic comparisons lead to a large delay in accessing files? As further algorithms are devised, this delay will increase...

2.1.4 Smart, near and exact identification scanning

Smart scanning was introduced to tackle complex viruses with mutation engines; these scanners would filter through the 'junk' code that is purposely placed by the virus and would focus on areas of the virus where these subroutines were not present. This improves the chances of detection and is particularly effective against macro, script and polymorphic viruses. Near identification and exact identification scanners are much more accurate versions of simple string

Commented [MG14]: Placed where? Would a word like "deposited" or "introduced" be better?

scanners in which the system can scan a file and if it detects an exact match, then it will continue to remove and repair but if there is only partial (near) identification then the system can refuse the file/virus the ability to decrypt as it may be a variant of one family of viruses. As a result, this is very effective against polymorphic and encrypted viruses.

As this is a variation of Algorithmic Scanning, it will have those Pros and Cons as well.

Advantages

- Is much more effective against both simple and complex viruses as it targets part of the viruses that are typically not encrypted. Thus it also makes for a cost effective method for companies.
- These scanners are much more likely to detect complex (polymorphic) viruses as they scan for specific routines rather than just scanning for inconsistencies.

Disadvantages

- Unlike generic scanners, these scanners will run a little slower as they are more specific.
- More system resource will be required as complex viruses with larger signatures will have to be mapped as the system is scanned.
- As more complex viruses develop, it is likely that this detection method will be circumvented and new more complex methods will need to replace them.
- Specific detection routines may rule out the chance to catch other viruses.

Commented [MG15]: resources

2.1.5 Sandboxing

Sandboxing is a new approach to tackling malware. It will present a virtual environment in which the suspect code can be placed in a virtual machine where mirrors of the programs and files of the local machine will be present. The code can read many of the files (even registry data) of the system and the virus is free to execute within the confines of the VM. When execution is finished, the changes in system files are logged and discarded. This would seem an ideal way to manage polymorphic viruses as changes in files and different virus addresses can be monitored and confined to the VM however this method also has problems.

Advantages

- It is effective against simple viruses; those without the capability to break the VM can be effectively monitored and logged.
- Simple viruses can read but not write to the local machine - this means that no files will actually be infected.

Disadvantages

- Many programs will not function as expected due to the restrictions of the network. This will result in compatibility problems and unreliable results.
- Complex viruses with advanced network capability may be able to exploit the VM and gain access to the local machine.
- As a relatively new system, the VM may have loopholes which viruses may be able to use, affecting the local machine instead.

2.1.6 Code Emulation

Much like Sandboxing, Code emulation utilises a VM to simulate code execution. The CPU and memory are mirrored in the VM and no actual execution takes place in the CPU itself. This is done to enable compatibility with all the programs being emulated. The CPU simulation will operate according to commands sent by the program, and this will loop for as many iterations as required by the software. Complex encrypted viruses such polymorphic can be detected by analysing the VM code after the iterations because polymorphic viruses tend to decrypt themselves after they have executed. The changes detected by string sequence scanners will be logged and the results discarded. An effective method of detecting complex viruses is a combination of code emulation and *dynamic-decryptor detection* (8). This involves using specific algorithms to detect the entry-point of the virus file and the VM to emulate said file, the specific encryption/decryption routines of the virus are noted and a profile is created. Furthermore, code can be optimised to remove the junk code of encrypted viruses. Thus it would seem that this is an effective technique in the detection of encrypted viruses.

Commented [MG16]: the said file

As this is a variation of Sandboxing, it will have those Pros and Cons as well.

Advantages

- The User is in complete control of the VM, most viruses can be contained and the user can see the results of an infection without any damage to the local machine.
- Dynamic decryptor detection combined with code emulation is a quick, effective method of removing the junk methods and exposing complex polymorphic viruses.

Disadvantages

- As every virus is different, it can be difficult to know when to stop the emulation, thus a complex virus may not fully decrypt and the VM may not detect it as a result.

2.2.0 Evaluation of the methods

In this section, I will evaluate all of the methods of detection above in terms of their capabilities in detecting polymorphic viruses, their practicality in doing so and how well I think they will stand against future viruses.

In general, all methods of detection of polymorphic viruses have had partial success however some have had more success than others. For example, sand boxing has had limited success when compared to algorithmic scanning methods. This is because Polymorphic viruses are naturally complex and the VM can withstand most simple viruses but the more complex viruses such as aggressive retro viruses have proven problematic; specific algorithms do not have this problem as they are simply scanning for viruses (not executing them). Furthermore, sand boxing is relatively inefficient when compared to specific variants of algorithmic scanners: smart scanners and near/exact identification methods take advantage of the 'unprotected' strings of the virus code. It is apparent that algorithmic methods are far superior to integrity checking in the sense that integrity checking has too many prerequisites such as a clean state and too many false positives. In addition, I feel that behaviour blocking, although effective, is too simple for the level of complexity that polymorphic viruses represent and requires a team of developers to 'be on the ball' in order to keep on top of the polymorphic threat. Code emulation, like smart scanning, can effectively remove the 'junk' code of a virus which polymorphic viruses create. The VM of code emulation is mostly secure and allows for viruses to execute and decrypt so that the virus itself is 'exposed': an ideal for scanners. As a result, I feel that code emulation and algorithmic scanners (with its variants) are the most effective methods of detecting viruses in use today and fair particularly well against polymorphic viruses.

Commented [MG17]: and can generate too

Commented [MG18]: decrypt themselves

This does not mean that these methods of detection are practical however. Both code emulation and sand boxing are particularly slow to run on machines as they must essentially run two copies of a machine in order to simulate the correct environment. Specific algorithmic scanners such as smart scanning are also rather slow. Previously, this had been a problem as only the developer had system resources capable of simulating a VM but with the rapid improvements in technology, this capability is commonplace in many systems now. One looks at the powerful Intel i5, i7 and AMD bulldozer CPUs which are commonplace in new technology. Of course VM emulators are slow in comparison to integrity checks. Integrity checking is a quick process as it simply has to compare the checksum of files and behaviour blocking simply flags anything suspicious. Behaviour blocking is not entirely practical as it must constantly monitor the system (taking up resources). Both integrity checking and behaviour blocking are easily defeated by complex, encrypted and memory resident viruses such as a polymorphic virus which can bypass detection. So, it would seem that speed (of detection) is a sacrifice of a little (time and memory) for a lot more (specific routines and a higher rate of detection). Therefore, code emulation and algorithmic scanners are the most practical and that generic scanners will, for the most part, be phased out with the growing presence of complex polymorphic viruses.

The complexity of viruses will increase so much so that they will pose a greater risk in the near future. This is because we store almost all of our information on computer systems today and it is likely that technology and our lives will be merged intensely as technology develops. Ray Kurzweil predicts that 'by 2019, a thousand dollars of computation will be equal to the most conservative estimates of the amount of computation we need to simulate the entire human brain' (6) and that

we will soon try to put computer systems in our brains (this is happening today with the hearing impaired – hearing devices) that can make everyday life much easier for us. As a result, extremely complex viruses are likely to emerge that will require the upmost attention to ensure that said systems are not affected. With this, it seems that simple and generic scanners, although they will advance with time, will quickly become obsolete in the face of the polymorphic threat. Furthermore, the present limitations of our computer systems will be lessened (if not removed) in the future, thus more complex options such as algorithmic methods and emulation seem much more feasible in real life. Of course, what will be considered 'simple' scanners in the future will work for the majority of viruses but, for the more complex viruses that could potentially threaten human life it would seem that algorithmic scanners, and not code emulation, would be best. This is because code emulators allow the system to execute the virus, and if the virus were complex (like a polymorphic), then it may break that emulator. Therefore, currently, it seems that algorithmic scanning methods and code emulators are the best form of detection for polymorphic viruses; when bio-technology becomes more commonplace in the future it is possible that code emulation will be phased out as to avoid any life-threatening problems.

3.0.0 Conclusion

In conclusion, I feel that the best methods of detection of polymorphic viruses today would be code emulators and complex algorithms such as smart scanners. Generic scanners do not fare well enough against the polymorphic variety and the only true way to detect an ever-changing polymorphic virus is to simulate its execution in order to expose it in raw, decrypted form. As time passes by, expectations are that the system limitations we face today will be removed; and with a little more development, code emulation will likely be circumvented so algorithmic scanners will be the most secure and efficient methods of detecting these threats. An unanswered question would be the extent and direction that technology will actually develop and how viruses will adapt but I feel that it is fair to say that the only effective way of detecting a polymorphic virus both now and in the future will be algorithmic or 'smart' detection.

Commented [MG19]: I would be tempted to reverse this final sentence, and the way it reads at the moment seems to finish the essay a little flat. How about : "fair to say that algorithmic or 'smart' detection will be the only effective way of detecting a polymorphic virus both now and in the future."

Bibliography

1. **Sky news.** FBI Server Shutdown Threatens Web Access. *Sky news.* [Online] Sky, 9 July 2012. [Cited: 11 July 2012.] http://news.sky.com/story/957795/fbi-server-shutdown-threatens-web-access.

2. **Kaspersky.** Review of the Virus.Win32.Virut.ce Malware Sample. *securelist.* [Online] Kaspersky. [Cited: 11 July 2012.] http://www.securelist.com/en/analysis/204792122/Review_of_the_Virus_Win32_Virut_ce_Malware_Sample.

3. **Pearson Education .** Computer Virus Timeline. *infoplease.com.* [Online] [Cited: 11 July 2012.] http://www.infoplease.com/ipa/A0872842.html.

4. **The British Computer Society.** Computer Abuse, Security and Related Law/Copyright. [book auth.] The British Computer Society. *A Glossary of computing terms, tenth edition.* s.l. : Addison-Wesley, 2002, pp. 109-121.

5. **Leyden, John.** PC virus celebrates 20th birthday. *The Register.* [Online] january 19 2006. [Cited: 11 July 2012.] http://www.theregister.co.uk/2006/01/19/pc_virus_at_20/.

6. **Kurweil, Raymond.** Ray Kurtzweil - futurist. [interv.] Dag Spicer. *futurist.* s.l. : Computer History Museum, 13 July 2009. The recording of the video was hosted on youtube.com (follow link).

7. **Symantec.** Understanding and Managing Polymorphic Viruses. *symantec.com.* [Online] Symantec, 27 September 1996. [Cited: 11 July 2012.] http://www.symantec.com/avcenter/reference/striker.pdf.

8. **Szor, Peter.** *The Art of Computer Virus Research and Defense.* s.l. : Addison Wesley, 2005. ISBN-10: 0321304543.

9. **Oxford University.** *Defintion of 'cyber terrorism' from Oxford English Dictionary.* Oxford : Oxford University, 2010. ISBN 0199571120.

10. **Landesman, Mary.** Boot-sector Viruses. *About.com.* [Online] [Cited: 12 July 2012.] http://antivirus.about.com/cs/tutorials/a/bsvirus.htm.

11. What Is Browser Hijacker Virus? How To Detect And Remove It. *combofix.org.* [Online] [Cited: 12 July 2012.] http://www.combofix.org/what-is-browser-hijacker-virus-how-to-detect-and-remove-it.php.

12. Understanding the Direct Action Virus. *spamlaws.com.* [Online] [Cited: 11 July 2012.] http://www.spamlaws.com/direct-action-virus.html.

13. macro virus. *webopedia.com.* [Online] [Cited: 11 July 2012.] http://www.webopedia.com/TERM/M/macro_virus.html.

14. **wikipedia.org.** Computer Virus. *wikipedia.org.* [Online] wikipedia, 21 April 2008. [Cited: 11 July 2012.] used to find the 'brain' virus example. http://en.wikipedia.org/wiki/File:Brain-virus.jpg.

15. **Dyer, Allan G M.Sc.** Boot Sector Viruses. *yuikee.com.* [Online] september 1 1994. [Cited: 13 July 2012.] http://articles.yuikee.com.hk/published/msm/msm01.html.

Appendix A *Relevant technical terminology*

- Bit (b): stands for binary digit, this is either a 1 or 0 in computing language or a true or false in Boolean.
- Byte (B): a standard unit of memory in computing, equivalent to 8 bits.
- Kilobyte (KB): a standard unit of memory in computing, equivalent to 1024 bytes.
- Megabyte (MB): 1024 Kilobytes.
- Gigabyte (GB): 1024 Megabytes.
- Terabyte (TB): 1024 Gigabytes.
- RAM, Random Access Memory, memory modules that store data for quick access and retrieval; temporary storage.
- ROM: read-only memory, memory modules (often in the CPU) which are available for instant access and retrieval, data cannot be written to this memory.
- CPU: the central processing unit; this is the 'brain' of the computer system; it reads writes and does calculations on data so it can be computable.
- HDD: Hard Disk Drive; permanent storage drive of which data can be written and read and removed. Operates via the motion of disks and tend to be in the 500MB-1TB size currently.
- SSD: solid state drives; a relatively new addition to data storage; it uses flash memory to read, write and remove data from flash drives. These are very fast but very expensive and tend to be in the 80GB-120 GB sizes currently.
- File: a collection of data; often intended to execute software.
- Data: a collection of values and code that do things when executed; measured using Bytes (*see above*).
- Malware: an abbreviation of the term 'malicious software'; used to describe programs that damage computer systems.
- Code Injection: an exploitation used by Cyber Terrorists, code is 'injected' into files in order to change order of execution.
- Recursion: the term used to describe when a method or function can call itself, a virus can use recursion to ensure that copies of the virus are in memory.
- Cyber Terrorism: the act of the 'politically motivated use of computers and information technology to cause severe disruption or widespread fear' (9).
- OS: abbreviation for 'Operating System'; a piece of software between the computer system and the user.
- Hardware; the physical components of a computer system such as RAM or a CPU.
- Software; the files and programs that a computer system uses to operate things such as applications. Windows 7 is a piece of software and also an Operating system.
- Peripherals- additional non-essential components of a computer system such as a mouse or speaker system.
- MS-Dos: abbreviation for Microsoft disk operating system. This is an old form of Operating system which was phased out by the more modern GUI used in most operating systems today.

- Boot Sector: a sector (or area) of a storage drive in which the fundamental commands and programs are stored which manage the booting of the software once a system has successfully reached POST.
- POST: acronym for power on self-test. This is a diagnostic test which is run each time a computer system is turned on to check if components are functioning correctly.
- Macros- a shortcut to a commonly used command in office software. It saves the user time.
- Checksum- a digit representing the total of the digits in stored or transmitted data.
- Payload: the contents of a file that is being transmitted, it is often the reason why said data is being transmitted.
- P-code: pre-compiled code, often used in object oriented programming languages. It is code that makes up a series of incomprehensible instructions that the CPU uses.
- False positives: when a system detects a change in the checksum of a file and issues a 'false alarm' as it assumes this is a virus' work.
- False negatives: opposite to false positives; it will check the checksum and not flag suspicious activity when it should do so.
- Digital signature- the binary string of characters from which they are constructed with each iteration.

Appendix B Virus definitions

Classification of viruses

In order for a virus to be able to operate, it must acquire permission from the system to execute the file and write to memory. In recent years, the complexity of computer systems has advanced so much so that this has proven rather difficult. As a result, cyber terrorists have designed code that utilises code injection in order to attach malicious code to files which will execute when the system passes control to the file in question. These viruses are often split into groups; these are resident and non-resident viruses:

Memory Resident Viruses

Memory resident viruses tend to have two functions: a finder module and a replication module. When the file containing the virus is executed, the payload will write a copy of the replication module to memory to the RAM so that the module is called and executed every time the OS requests an operation. They tend to be classified further as fast or slow infectors. The fast infectors will try and infect every eligible file as quickly as possible but risks detection from anti-virus software. A slow infector will occasionally infect files as to avoid this (every time a file is moved for example) but tend to be ineffective.

Non-resident viruses

Typically non-resident viruses also have two functions: a finder module and a replication module. The finder function will search the drive for potential host files, once one is found the replication module will use methods such as recursion (and polymorphism) in order to infect said files. These viruses do not maintain residence in memory. Typical targets include executable (.exe) files and command files (.COM) in MS-Dos and Extensible Linking Formats (ELF) in Linux.

Of which can be categorised into 7 main types:

Boot Sector viruses

Boot sectors infect the operating system file in the boot sector of the drive, potentially putting the boot sector at risk and they could potentially stop the OS from running. (10)

Browser High jacking viruses

A memory resident program that will block the standard home and error pages of the user's internet browser, it will redirect the user to its own webpage where there is often more malware waiting to infect the computer. (11)

Direct Action Viruses

A memory resident virus that uses selective code to determine which file is infected each time the program executes. This is a serious threat as normal operation can be seriously affected. (12)

Macro Viruses

Many pieces of software use macros such as word processors and spread sheet editors, macro viruses are written in the respective language of its intended software; the virus is embedded into the document and is executed with each successive opening of said software or file. (13)

Multipartite Viruses

This is a fast acting non-resident virus that attacks both the boot sectors of drives and the executable files. This makes for a potential system crippling virus.

Polymorphic Viruses

These are stealth viruses that affect the respective files and use encrypted polymorphism to recreate altered versions of the same virus. This potentially means that its pattern is hard to detect.

Trojan Viruses

Much like the Trojan horse of roman history; the Trojan virus will 'perform a normal process in the computer but will also perform another, possibly harmful, process at the same time.' (4).