The International Journal of Multi-Disciplinary Research
ISSN: 3471-7102

ONCE OFF VIRUS SCANNER

(Conference ID: CFP/347/2017)

By
Kalasile chaila
kalasilechaila@ymail.com
Engineering Department
Information and Communication University
Ndola, Zambia

Abstract

Computer viruses pose a considerable problem for users of personal computers. The recent emergence of macro viruses as a problem of some importance may heighten virus awareness in general. Yet most people have little or no understanding of common virus scanners, the varieties of viruses that exist today, and the strategies which they use to accomplish infection and to defeat scanners or anti-viruses. It is well-known that the virus problem is most severe for users of IBM PCs and compatibles; however, users of other platforms, such as the Macintosh, should not become complacent – viruses exist for many platforms in varying numbers. The ease with which macro viruses may be written is discussed and a new virus attack for the Macintosh is presented which closely resembles an attack under DOS for the PC. Thus, this once off virus scanner is vital in order to eliminate and identified these malicious viruses that been created every day. This study will outline how this scanner work and why it is very good virus scanner.

Keywords: *computer virus, macro viruses, virus scanner, computer science, companion viruses.*

1. Introduction

1.1. Background

Computers and computer users are under assault by hackers like never before, but computer viruses are almost as old as electronic computers themselves. Most people use the term “computer virus” to refer to all malicious software, which we call malware. Computer Viruses are actually just one type of malware, self-replicating programs designed to spread itself from computer to computer. A virus is, in fact, the earliest known malware invented.

There are many methodologies that can be employed to come up with a software product or solution such as Tradition life cycle, prototyping, spiral and more. Each of these methodologies has its own pros and cons. They are applied to a solution depending on their suitability.

They are various types of models are;

- Waterfall Model
- V-Shaped Model
- Evolutionary Prototyping Model
- Spiral Method (SDM)
- Iterative and Incremental Method
- Extreme programming (Agile development)

The **Once off virus Scanner** employees the waterfall model and evolutionary prototype system development because firstly the technology used is not dynamic, it’s a short project, it is a project that should be reusable and simple to understand hence the use of waterfall model and the prototype way of developing systems will best suit the development of the actual system. Once off will work with the evolutionary prototype system development so that once finished it will become the actual system being developed.
In the water fall model, we see very well the pros and cons below;

Advantages	Disadvantages
▪ Easy to explain to the users. ▪ Structures approach. ▪ Stages and activities are well defined. ▪ Helps to plan and schedule the project. ▪ Verification at each stage ensures early detection of errors / misunderstanding. ▪ Each phase has specific deliverables.	▪ Assumes that the requirements of a system can be frozen. ▪ Very difficult to go back to any stage after it finished. ▪ A little flexibility and adjusting scope is difficult and expensive. ▪ Costly and required more time, in addition to the detailed plan.

And similar in the prototype system development we see below the pros and cons;

Advantages	Disadvantages
▪ Reduced time and costs, but this can be disadvantage if the developer loses time in developing the prototypes. ▪ Improved and increased user involvement.	▪ Insufficient analysis· User confusion of prototype and finished system. ▪ Developer misunderstanding of user objectives. ▪ Excessive development time of the prototype. ▪ Expense of implementing prototyping

Waterfall model explained

Waterfall model works well in the sense that it allows the developer to finished one phase before you can go over to the next this enhances documentation for easy understandability of the system and allows for well-planned design of a system in this case once of scan.

Once off requirements and documentation

In this stage developer put together the entire requirements Product manager creates requirements documents that include the following requirements e.g.

- User should be able to learn in short time frame
- User should be able to operate with minimal change in the main IT system

Analysis of once off virus scanner

The software development team analyzes the requirements, and fully understands the problems. This is a research phase that includes no building. The team attempts to ask all the questions and secure all the answers they need to build the security requirement.

Design of once off virus scanner

The software developers design a technical solution to the problems set out by the system requirements, including scenarios, layouts and data models. This phase is usually accompanied by documentation for each requirement, which enables other members of the team to review it for validation.

Development of once off virus scanner

Once of will use the evolutionary way of developing to allow flexibility of the programmer and also to enable him be able to alter and check to see if the alterations are valid this will enable the user be able to complete the program in a shortest time and still deliver best results

Testing of once off virus scanner

After have been developed once off will have to go throw a series of actual tests to see that it achieves the goals that where set in the planning

Once off virus scanner implementation

Upon the testing having been carried out the developers can go ahead and implement it using anyone of the various implementation processes.

1.2. Problem Statement

Any computer can be infected by malware. Malware is a catch-all term for malicious programs, such as viruses, worms, Trojans, and spyware, which are designed to infect and take control of a computer. Once your computer has been infected, bad programmers can capture all your keystrokes, steal your documents, and use your computer to attack other computers. It is in this light the researcher thought of creating a cheap scanning tool that can help computer users protect themselves from computer viruses.

1.3. Objectives

- To provide last mile gate keeping security
- To help an organization keep sensitive information free from viruses
- To provide cheap scan tool by virtue of it being a freeware
-

1.4. Research questions

i) What is a computer virus?
ii) How are computer viruses created?
iii) Why do computer users need virus scanners?
iv) What could be the best way of detecting a virus and securing files and documents?

1.5. Significance

Once off Scan virus cleaning tool provides an opportunity to verify or check files for viruses before they are stored by preserving information safely.

Resource wise it demands less of an organization IT infrastructure or resources in the sense that it does not change how the system works or does not need you to alter how the system works for it to run. Simple to use and access in that it is a freeware.

2. Literature Review

2.1. What is a computer virus?

There is some difficulty in producing a definition for the term “computer virus”. Dr Cohen has presented a mathematical definition of a computer virus, which may be roughly expressed as: A virus is a program that can ‘infect’ other programs by modifying them to include a possibly evolved version of it [4]. However, this definition classifies as viruses many things which would not be considered viruses by those working in the anti-virus field. At the same time this definition would not consider as viruses programs that infect another without modifying the target program itself [2] (examples of such viruses are the companion viruses, discussed in section 2.3).

Additionally, the above definition does not convey any need for the “virus” to be able to replicate further once it has infected some other program [4] – and replication is viewed as an important characteristic of a true “computer virus”.

A definition which is felt to be more practically useful when dealing with real computer viruses than Dr Cohen’s mathematical model is: we define a computer virus as self-replicating program that can infect other programs by modifying them or their environment such that a call to an infected program implies a call to a possibly evolved, and in most cases, functionally similar copy of the virus [21, 4].

The term infect is used with respect to computer viruses in the sense of the definition above throughout the remainder of this study. It is important to note that a virus is not necessarily malicious although it may have side effects as a result of the virus clashing with the operating system, user programs and extensions to the standard operating system installed by the user which are deemed to be undesirable.

Some researchers have been considering the question of the viruses that perform useful cations so-called benevolent viruses. Cohen [4, pp 15-21] considers briefly the topic of benevolent viruses. He considers as examples:

Comprehension viruses – little-used files are compressed by the virus and uncompressed when required.

Maintenance viruses – any virus which would perform maintenance tasks in a computer system, such as updating installed programs.

Distributed database with viruses – viruses would produce on networked computers, performing searches for the virus' originator. Results would be reported back by mail, and the virus would clean itself up after a certain time.

The use of viruses in covert distributed data possessing (specifically, key cracking on encrypted messages) has been proposed by White [13]. It may be argued however, that this would not be particularly beneficial to the user whose resources are used by the virus.
Many arguments against the idea of benevolent viruses are presented by Bontchev [2]

Worms, Trojan Horses, Droppers and Logic Bombs
A worm is an independent program that is able to spread copies of itself or part of itself to other computers, commonly across network connections, and these copies are themselves fully functional independent programs, which are capable of either spreading further and/ or of communicating with the parent worm (to report back the results of some computation, for example).

There is a common confusion over the distinction a **worm** and a **virus**. For example, the program that negatively affected the internet 1988 is referred to as a virus ("the internet virus") by some [5] and as a worm ("the internet worm") by others [11, 12, & 10]. Spafford [12] argues that referring to the infection as "worm" rather than a "virus" is most appropriate.

A notable difference between worm programs (such as the internet worm) and viruses is that while a virus may take advantage of network connections to infect other programs (some local area networks are particularly susceptible, as the user is able to interact with programs and data stored on a remote machine. Clearly the well-known worms have been able to cause their programs to be executed on the remote machine which was the worm's target.

The internet worm affected Sun 3 and VAX systems running variants of BSD UNIX [11]. Other worms have been created with other networks in mind, such as DECnet [7]

A Trojan horse is a program which possesses various intentional undocumented features to manifest themselves. Unlike a computer virus, which attaches itself to some other program using any a number of methods, a Trojan horse is a self-contained program. A Trojan may have functions to use to the user. Some definition of "Trojan horse" defines a computer virus as a replicating Trojan horse which inserts a copy of itself into other program [1].

A Trojan horse might install a virus as its intentional undocumented action. Some feel that a program which installs a virus as result of having been previously infected by a virus is also a Trojan horse (having been converted to Trojan horse by the virus) [18]. However, this seems incompatible with the notion of a "Trojan horse" being a program which was initially produced with an international undocumented feature in place.

A dropper is a program which acts as a carrier for a computer virus. A dropper is not the result of a normal infection of some program by a virus – it exists only to spread the virus. The virus is usually kept by the dropper in a form which will not be detected by anti-virus software [18]. Some macro viruses (see section 3) attempt to act as dropper for more conventional varieties of viruses [15], in addition to other actions they perform.

Varieties of viruses

There are a number of different ways that viruses use to infect the computer system. The two main types of viruses are:

File infectors: these are viruses that attach themselves to some form of executable code. There is a variety of ways in which a virus will attempt to infect a file. On a DOS-based system, file infectors will commonly attach themselves to .COM or .EXE files, although many other kids of infect able objects.

Boot sector infectors: only discussed in the context of a PC-compatible system. These kinds of viruses infect executable code which is loaded from the disk and called when a computer is starting up. There are a number of different pieces of code which may be modified by a virus to infect a system, such as:

- DOS boot sector [floppy disks and hard disks]
- Master boot record (MBR) [hard disks].
- Partition table [hard disks only]

A virus that is capable of spreading by infecting files and by infecting via any code executed at boot time is known as **multipartite** virus.

Boot sector virus is extremely widespread; as a group they are easily the most found variety virus on PC- compatible systems.

A virus may be **a direct-action or resident** [18]. A direct-action virus is one that when initially executed in the course of normal use of a computer system identifies executable objects for infection and exits once infection has been accomplished.

Direct-action virus may be referred to as non-resident viruses. A resident virus is one which installs itself somewhere in memory, and makes arrangements for the body in memory to be executed at some future time; the virus may infect files or take other action (to conceal its presence, for example) at the time it is next executed. For example, Macintosh viruses if resident in memory will infect an application when that application commences running and perform certain systems calls that initialize the Macintosh Toolbox, which consists of asset of utility functions available to all applications.

Some programs useful to the user are also resident programs- this includes some antivirus programs that monitor computer system operation for actions which may indicate the presence of a virus. There are some other types of viruses which should be mentioned:
Macro viruses: these are explained in detail in section 4 the ease with which such virus may be written is discussed in section 9.2.

File system or Cluster Viruses: rather than infecting the files directly, such a virus modifies directory table information so that the virus is executed first. It would then pass control to the program that the caller really wants so as to avoid rapid detection. “Dir-II” is an example of his variety of virus [18].

Kernel viruses: these are viruses that target specific features of an operating system’s Kernel (the programs) that represents the heart of an operating system [18].

Companion viruses: companion viruses occur in several varieties [3, 8], the most notable being:
Regular companion: creates a file in the same directory as the target of infection but with a filename extension which operating system chooses to execute before the original file (for example, under DOS a. COM file is executed before a .EXE file with the same name). This would appear to be an attack which is highly specific to PCs running DOS.

PATH companion: create a file any executable extension in the directory that is searched for executable files before the directory containing the target infection (named after the PATH environment variables found in operating systems such as DOS and UNIX).

Obviously, not all of these infection strategies are available on some platforms and operating systems. For example, Macintosh computers do not suffer from companion viruses as described above in any form, and also don’t appear to be afflicted with viruses of the boot sector variety. All viruses, with an exception of macro viruses, are platform dependent.

Macro viruses

The anti-virus community has been aware for some time now of the potential for virus writing provided by the scripting (or macro) languages of large software packages such as Microsoft word and excel. The idea of macro viruses was first introduced by Highland [6].

However, it is only recently that such viruses have become a problem. These viruses are remarkable for the fact that they infect what is thought of as documents. A macro virus may also be platform independent, being capable of spreading on any computer platform supported by the host application. The most well-known and wide spread are Microsoft word viruses written in WordBasic (two examples are concept and nuclear).
Microsoft word recognizes the existence of two different forms of user file – an ordinary document, and template. A template is much more like an ordinary document, with the addition that templates may also contain macros written in WordBasic. Ordinary documents may be converted easily into templates, but the reverse is not the case. Word has a “global template”, the so-called the “normal template”, which is used by every document created, and is important for the spread of macro viruses – the macro that make up the virus are copied into the normal template where they are subsequently available to other documents.

Word macros may be marked as executive only, which means that the macros cannot be easily edited or inspected by a word user but only be executed. Some viruses for example, the nuclear virus, use this method to hinder casual analysis of the viral macros.

Macro viruses in Microsoft Word exploit the existence of several varieties within the Word-basic [15]:

- **The AutoExec** macro, stored within some global template such as the Normal template, which is executed automatically whenever Word is started.
- “Auto” macros, which run whenever certain user take place within the Word:
    - **Auto New** runs when a new document is created.
    - **AutOpen** runs when an exciting document opens.
    - **autoClose** run when an open document is closed.
    - **AutoExit** runs before exits when the user quits.

- Macros named for Word menu options, which are run when the menu option is selected
  A basic Word virus is not difficult to create, requiring just one macro. The macro virus **DMV** features a single macro **AutoClose.** The Normal template will be infected when the document containing the DMV AutoClose macro is closed. Subsequently, documents are infected as they are closed [15]. **Concept,** a more complicated macro virus, signals its

initial infection of the Normal template, and subsequently will infect any document with the viral macros when the document is saved using "the save as..." option of the "file" menu (an example of a virus using a macro named for a Word menu option).

Some macro viruses, such as DMV and concept do nothing but spread. Some would argue that even this is damaging, in terms of time required to remove the virus macros in whatever documents have been infected. More damaging actions are certainly possible, however. For example, a virus might delete paragraphs from a document. Or insert words (the nuclear virus will append some lines of text to documents when printed at a certain time). More sophisticated macros viruses attempt to infect the users with an ordinary variety virus which infects the executable files (the Nuclear virus unsuccessfully attempts to do this), however, behavior such as this platform dependent.

A virus can also give itself some "limited" protection against being removed by implementing a **Tools Macro**, which is then executed in place of the corresponding menu option should be selected by the user. This menu command offers one possible way for macros to be removed from a document, if the environment has already been infected by a virus.
Disabling the features of Microsoft Word which allow execution of certain macros when some documents are opened and closed offers some limited protection, but as macro viruses can be written with macros that mask menu options (macros which cannot be disabled in the same way that automatic macros may), this is hardly a complete solution.

Virus occurrences

There have a great many viruses created for many different computer platforms. This PC has by far largest share of all viruses in existence (the producers of Dr. Solomon's Anti-virus Toolkit, a leading anti-virus package, claim to detect 9417 PC viruses [20]).

Many of these PC viruses are closely related (once a virus becomes available, it is not uncommon to find a number of copycat viruses that differ only slightly from the original appearing, perhaps written by less-skilled virus writers using virus source code; alternatively, the virus with different payloads but sharing common code foe infection and anti-virus measures).

Some information has been gathered by virus bulletin about PC viruses that have been reported as found over the course of the month for some months. The percentage of the percentage of the reports made up by the various virus classes for several months of 1996 shown in the table below:

	Jun	Jul	Aug	Sep
macro	21.8	18.4	20.2	34.3
boot	62.5	64.5	59.2	53.7
multipartite	7.3	10.6	13.9	6.6
file	8.4	4.8	6.4	4.8
other	0.0	0.3	0.0	0.0
unclassified	0.0	1.3	0.6	0 .6

The table shows the reports to/ collected by virus bulletin made various classes of virus.

Only a few the total numbers of known viruses is responsible for the majority of virus incidents. Some on-access anti-virus products use this fact to help restrict the number of viruses that a file must be checked for when accessed by a user.

Viruses exists for number of other personal platforms, such as the Amiga and the Macintosh, but the numbers of these viruses are a small fraction of the numbers of viruses available for the PC the Macintosh, for example, is afflicted only a few dozen viruses. Viruses written to target UNIX are especially uncommon.

Viruses Anti-Detection / Anti-Analysis Strategies

There are a variety of strategies which a virus might employ to hinder detection of the virus, and analysis once I has been discovered:

Stealth: While the virus is active in memory, it can intercept disk reads, and when it detects and attempt to read section of the disk (such as a boot sector, partition table or master boot record) or a file that has been infected by the virus, can conceal its presence by returning as a result of the call data with no signs of infection. Most stealth viruses are boot sector viruses (it being much easier to detect reads of these areas of the disk). Some examples of stealth viruses are 17]:

- Members of the “brain” strain of viruses are stealth floppy boot sector viruses.
- The “512” virus (also known as “Number of the Beast”) is a stealth file infecting virus.

Polymorphic viruses: a polymorphic virus is one that is capable of varying many aspects of its appearance in the hope of avoiding detection. Cohen refers to viruses of this type as “evolutionary” viruses [4, p.73]. the strategies which might be employed by the polymorphic are many of these strategies for code ‘evolution’ ea explained in greater detail by Cohen [4, pp. 199-215]:

Encryption: encrypt the body of the virus not only with the variety of different keys but with the variety of different encryption strategies (each requiring a different descriptor, of course). The encryption approach is a particularly common polymorphic trick – its wide use was facilitated by the distribution of the variety of object code modules (such as MtE and TPE) which could make any virus polymorphic. Additionally, this is the most straight forward polymorphic strategy which a virus might employ. One of many viruses which employ a strategy of this sort is the “tequila” virus [17].

Instruction Equivalence: some machine instructions achieve equivalent effects. Substitute for these equivalent instructions in the virus code.

Equivalent Instruction Sequences: replace one sequence of machine instructions with another which achieves the same final result.

Instruction reordering: sometimes it is possible to reorder a sequence of instructions in a variety of different ways and still achieve the same result. The “1260” virus is one which alters the order of instructions in its decryption routine from infection to infection, and additionally inserts irrelevant instruction [17].

Variable Substitutions: alter which memory location contains each of the variables used by the program. Especially effective if the variables are dispersed about the program.

Add or Remove Jumps: add unnecessary jump/branch instructions to program while still preserving its function (this makes the program longer, of course), or remove the system jump/branch instructions by moving the program code which was the destination of jump/branch.

Add or Remove Calls: replace calls to subroutines with a copy of subroutine; replace the sequence code with a subroutine call.
Garbage Insertion: foe example, might add NOP instructions, or other instructions which do not otherwise achieve any useful purpose apart from obscuring the code’s functions. The “tequila” virus employs a strategy of this sort [17].

Simulation: replace an instruction sequence with an alternative sequence which achieves the same effect when interpreted (or simulated) appropriately.

Build and Execute: “build” instructions somewhere in memory (one byte or a sequence of bytes at a time) and then execute them. This obscure the instructions being constructed from observation until their execution.

Intermixing Programs: interleave the instructions for separate programs or blocks or code in such a way that the interleaved code achieves the same result as the separate blocks of code.

Tunneling: a technique sometimes used by viruses that attempts to bypass activity monitoring virus detectors (for example, by bypassing the operating system disk access functions and interpreting the contents of the disk itself, or calling the operating system's functions directly to avoid any trap the activity monitor may have set in place) or otherwise subvert anti-virus techniques and strategies.

Cavity Virus: a variety of file affecting the virus which overwrites a portion of the file which is filled with the same value, such as region filled with zero bytes. Such a virus will not alter the total length of the file.

Ant debugger Mutation: various tricks can be employed to make dissembling and tracing the program code difficult. For example, the program code might be arranged in such a way that an instruction code set of instruction or set of instructions is hidden from casual inspection, so that the function of the code is no longer readily apparent from its disassembly. The debugger itself could not be manipulated in the course of tracing a program by tampering with the debugger's address space.

The speed with which a virus produces and infects other files is also important. Some viruses spread particularly quickly. For example, a virus might be programmed to infect executable file when the file is opened, for whatever reason. These are so-called fast infectors.
A slow infector is virus which only infects the file as they are created, immediately prior to following a legitimate change (that is, some change that the user wants to have happened), or as files are copied, perhaps onto a floppy disk [3, 9, 18] [4, p. 91].

Defenses Against and Detection for Viruses

There are a variety of defenses against viruses, and ways of detecting their presence. A natural first question is: "is it possible to detect all viruses?". Unfortunately, not- it is mentioned in Cohen [4, pp. 64-68] that it is not possible to extract some program which correctly determines whether or not some other program a virus. In fact, given a known virus, it isn't even possible to systematically using a computer program if another program is affected with the virus derived from the original known virus in some way.
However, there are a variety of impact ways to detect the (possible) presence of computer viruses. Most forms of virus detections involve the false positives, which occur when object is identified as being infected by a virus when in fact it is clean, and false negatives, which occur

when an object is passed as clean when in fact it is infected with a virus. Ideally, false positives and false negatives will occur very infrequently.

Some techniques such as scanning for viruses, often lead to positive identification of a virus. In many cases, the infection (and some damage) caused by a virus is reversible.

Stealth techniques mean that attempts at virus detection that manipulating file objects on disk will not necessarily be effective if the virus is present and active in memory. For this reason it is recommended that before attempting to detect a virus, the computer in question is rebooted from an infected, locked, floppy disk containing clean version of whatever anti-virus softwrare is needed to search for viruses.

Known-Virus Scanning: attempts to identify viruses by scanning file for certain strings of bytes known to occur in particular viruses. Simple scanners which perform only such searches are easily defeat able by a well able and sophisticated polymorphic virus, so many also employ some more advanced techniques (such as heuristics analysis, to detect suspicious code fragments, or algorithmic analysis, to detect complex polymorphic viruses). Scanners often have difficulties detecting new viruses, and they require frequent updating.

Heuristic Analysis: attempt to identify possible virus by looking for codes that perform functions which in combination with each other are deemed to be suspicious. An example of a suspicious code fragment would one that alters the first few bytes of an executable file in memory- this would be require so that an infected executable could run normally when infected with a virus.

Behavior Blocker / Monitor: attempts to detect viruses based on patterns of virus activity. This approach has problems because many of the actions performed by a virus are perfectly legitimate under other circumstances. These methods are also sometimes rendered ineffective if a tunneling virus is infecting the system – these frequencies are frequently able to bypass the methods used by a monitor to detect virus activities.

Integrity Checker / Integrity Shell: integrity checking involves collecting a database of signatures for each file which is likely to be the target of the virus infection (such as, application programs). If at a later date this signature can be determined to have changed, then it is possible that the virus infection has taken place. This method will not detect viruses before infection takes place. There are a variety of enhancements to the basic method outlined here which must be implemented (for example, a companion virus does not necessary modify the item it "infects". So integrity checkers must attempt o identify the presence of a companion virus by other means).

An integrity shell [4, pp. 83-93] is sophisticated approaches which involves checking every object on which some object X (which the user wishes to use in some way) depends.
An integrity checker is generic anti-virus program-it is not targeted at a specific virus or class of viruses and so will rarely need updating. Integrity checking issues are extensively discussed in papers by Bontchev [3] and Radai [9].
There are a variety of other ways to help prevent virus infections. For example, as great majority of PC viruses are boot sector a virus, changing the order in which the computer uses its disk drives for a bootable is an effective defiance in many cases (an apparently common setting is to attempt to boot a floppy disk before attempting to boot a hard disk). Precautions will still have to be taken to deal with multipartite and other non-boot sector viruses, of course.
Cohen outlines a number of strategies that will prevent or hinder computer viruses from spreading throughout the computer system or computer network [4 pp. 57-64]. The most interesting of these approaches is that of limited sharing. The best that can be done to limit sharing is transitive information networks that implement sharing is to base the structure on a "partially ordered set", or POset. This means that information can flow in only one direction, for example, from host A to host B but not from host B to host A. this effectively limits the possible range of viral infection, and also helps to trace the origin of any suspected infection, as the souse of infection would be one of the numbers of machines on the infection was ultimately.

The use of a vaccine against a certain computer virus is a technique no longer widely practiced. Most viruses check a potential infection target to make sure it is not infected by that type of a virus (to prevent multiple infections), so infection by a particular virus could be prevented marking executables so they appeared to be already infected (hence the name "vaccine"). The large number of viruses and the fact that some virus' identification techniques are mutually contradictory means that this technique is no longer workable.

Problems with and Attacks against – Viruses Measures

Virus scanners need updating with great frequency because of the speed with which new viruses are created and released. They are popular for number of reasons:

- A scanner is usually straight forward to use. Whether or not it is in recommended manner or is another matter.
- When a virus is detected it can pensively identified, and many scanners include facilities for "disinfecting" infected files.
- A scanner is the most reliable means of detecting a known virus in a new file; other techniques are not necessarily applicable (for example, an integrity checker cannot be used to check a newly-obtained program for viruses, because there is no way of determining what the signature of uninfected program should be).

A scanner will sometimes perform poorly when attempting to detect unknown viruses.

Polymorphism was at one time an effective attack against scanners merely searched for strings of bytes known to characterize certain viruses Cohen states that “until several years after the MtE was spreading in the world, no scanner was able to pick up over 95% infections” [4]. The situation has improved greatly in the recent times- most good scanners are capable of detecting the majority of polymorphic viruses.

As mentioned in section 7, stealth viruses may cause problems if the virus is present in memory using a virus scanner or integrity checker, as files presented for inspections may appear clean when in fact they are not. Furthermore, a variety of fast infector stealth virus may take the opportunity presented by numerous files for checking to infect those files; this presents a serious cleanup in an environment with hundreds (or possibly even thousands) of executable files. An attempt must be made to identify such viruses in memory.

Activity monitoring programs require updating as well, to cope with new viruses’ behaviors. There are some varieties of virus behavior detectable by a monitor, - such infecting only files which are about to be modified in any case. One particular virus, the “Darth Vader” virus, was designed to avoid alerting an activity monitor program by attaching itself to certain copies as they were copied [3]. This also presented an effective attack against integrity checkers at that time.

Activity monitors have the additional problems in that they may flag legitimate uses as well. They might also be bypassed by a tunneling virus or disabled in memory [3, 9].
Slow infectors are a concern for integrity checkers software. The virus infection may go unnoticed, because the integrity checker doesn’t have any signature in its database with which to compare that of the new file. An example of a common legitimate change to an excitable file is the addition of patches to the file by an updater program. Radai [9] suggest that it would be possible to detect the presence of slow viruses by creating a series of small executable files in the hope that one would be infected by slow viruses. Copies of the created executable could be created for easy comparison with the first this will no establish which other newly created or modified objects are infected with the slow virus, however. Tracking down the source of the infection could not be problematic.

Programs which cause modifications to their own executable code will cause problems for integrity checkers. It is difficult to know how such programming habits are practiced at this time. Integrity checkers will not be effective to all types of viruses. As integrity checking is usually applied only to hard disks (its application to floppy disks is not practical, as the contents of the floppy disks are frequently modified) a virus which infects floppy disks only and ignores hard disks would go unnoticed [9, 3]. The “brain” virus, an early DOS virus, is an example of virus which ignores hard disks.

Recent work

Companion viruses and the Macintosh

Macintosh viruses infect application files, the system file (a file present in every bootable Macintosh disk which holds many resources used by the operating system) or system extensions (small files containing executable code which load every time the Macintosh is started up and which add extra functionality to the operating system). A very few viruses which no longer work under recent releases of the operating system infected the Macintosh by more unusual means.
The Macintosh does not have any features which correspond to the preference of DOS to execute .COM files before .EXE files of the same name (when the selection is not explicit). Nor does it feature the concept of the "path" along the operating system searches for applications to execute if the desired application is not in the directory, as DOS does. So, a companion virus which does not alter the target application is not implementable in the same manner as under DOS.

However, it is possible to produce as virus with many of the same characteristics as a companion virus by manipulating a Macintosh disk's desktop database. This attack seems not to have been explored to date.

Macintosh files have both a **file type** (for example, **'TEXT** 'denotes a plain text file) and a **creator**. Each application should have a unique creator code, with which the files that the application creates are marked. The "Finder" (the part of the Macintosh operating system which is responsible for presenting the graphical user interface and managing user interactions) stores information about the file creators which allows to determine which application should be launched when a document icon is double clicked, and what icon should have displayed for a file of a certain type and creator.

When there is more than one application present in the same creator, then the Finder launches the application with the most recent creation date when documents are double-clicked. Application which are launched as a result of double-clicking a document icon are sent by operating system what is referred to as a high-level **event** or **apple event**, detailing which document or documents are to be manipulated.

A viral application, with a more recent creation date than the infected application, would be launched by the Finder before the target application, and would then have an opportunity to infect other applications (for example, any application currently running which not already been infected, or perhaps by scanning the directory tree for uninfected applications, processing only a few directories at a time so as detection), to perform other actions. The viral application would then pass control to the infected application. The Apple Event which details the documents to be processed can easily be passed on to the target application, so that there is little outward evidence that anything unusual has taken place.

This method of infection seems to function well on hard drives with single and multiple partitions, and should infect Macintosh file servers as well.
The ability of such a virus to conceal itself is limited, being restricted to setting the position of its document icon displayed by the Finder to some point off the edge of the display window.
Like companion viruses of DOS, this attack avoids altering any exsiting executable code or system resources.

Macro Viruses
A traditional virus can be difficult to quit, difficult to write, as there are many factors which can be considered if the virus is to work successfully and be able to avoid detection on wide range of systems running a variety of software products.
Simple macro viruses, however, are not hampered by many of very simple macro viruses were implemented by Horton. A first attempt required only a single macro, **Auto-Close**. This attempt was very crude and would be easily detected by an alert user, as infecting a document when it was closed (causing the **Auto-Close** macro to run) required that the document be saved to disk with the viral macro attached.
A later more slightly sophisticated offering was implemented using two macros, **Auto-Open** and **FileSaveAS**. The AutoOpen macro would be activated whenever an infected document is opened in word. If Word's Normal template is not already infected, the virus then infects the Normal template, which is a global macro file visible by all documents, by copying the viral macros to the template.

From this point on, whenever the user selects **Save As…** from the **file** menu, the **leSaveAfis** macro (now attached to the Normal template) is activated instead. It performs the same tasks as the usual menu option, with the exception that before the document is saved it is converted to a template to which the viral macros are then attached.

Neither of the two macro viruses created implemented any "payload" macros, although the addition of a simple payload would be trivial. The viruses are not remarkable for the techniques they employ, which are found in other macro viruses.

The first implementation of the macro virus was accomplished in approximately eight hours by a computer science graduate previously unfamiliar with Microsoft Word and WordBasic. No documentation was available other than the "help" system of Microsoft Word and a freely available description of some common macro viruses [15].

The second attempt required about an hour, and was sufficiently sophisticated that it would be able to replicate unnoticed. A grater familiarity with WordBasic would be required to write anything more sophisticated, but the exercise illustrates the fact that macro viruses, because of their ease construction, permit a much wider pool of computer users to write computer viruses in a very short space of time.

Macro viruses are quite widespread and so samples are often easy to obtain. Furthermore, understanding the code of a micro virus (assuming that the macros are not marked as ExecuteOnly) is very much less difficult than interpreting the assembly language in which non-macro viruses are typically implemented. The study of techniques used to implement other macro viruses is one straight forward method of improving virus writing.

More sophisticated macro viruses might attach specific computer systems by attempting to "drop" a non-macro filesystem virus into the target system, or perform other operating system dependent actions. This would require greater familiarity with the WordBasic or other macro languages than are more demonstrated in a simple macro virus, such as the ones created as a part of this exercise.

Summary

This paper has presented a definition of viruses as the term is commonly used, as well as attempting to explain the meaning of some other terms such a "worm", which are often closely associated with viruses. The various types of viruses that currently exist and various methods used by those viruses for infection have been explained, and the types and numbers of viruses likely to be found in the wild have been considered.

The various ways viruses currently use to hider detection and delay analysis once they have been detected were discussed. Commonly available anti-virus measures were explained, as well as faults and problems that these methods are known to have.

Finally, some recent work on the ease of implementation of simple macro virus-type attack for the Macintosh was discussed.

3. Research, Design and Methodology

3.1. Qualitative Research

The literature review and its critical analysis in the chapters above have shown evidence that creating a once off virus scanner is of important. The implementation of this research project will be of great value to computer users who have important files and documents that need severe security. As shown above viruses can infect any computer, once off virus scanner is a very much needed virus detector.

3.2. Data collection methods

In order to gather data for this project, a variety of data collection tools are employed, such as in-depth interviews, literature review and observation (field notes). The rationale for using a variety of tools, according to Merian, quoted in Simon-uguru (1991, p.39), is that, the weakness of one tool is the strengths of another. Thus, by combining investigation tools, the researcher will be able to achieve the best of each, while overcoming the unique deficiencies of each. The researcher chose these instruments in order to help achieve the intended aim set for this project. Ghosh (1992, p: 213) asserts that "the relevance of using particular research instruments will depend on the aim of the study being carried out". For instance, if the aim is to understand how certain phenomenon are done, then the use of interviews and observation (field note) to collect the intended data is necessary. Thus, the researcher is using these tools in this project.

3.2.1. In-depth-interviews

An in-depth-interview, sometimes called information conversation interview is a direct verbal technique for obtaining data. It is a commonly used method of data collection in the study of human behavior of perception (Ghosh, 1992). In a qualitative survey, the main purpose of the interviews is to obtain a specific kind of information which the researcher may wish to find out. Partons (1980, p: 196) explains, "We interview people to find out from them those things we cannot directly observe such as feelings, thoughts, and intentions." Therefore, the purpose of the interview in this project is to allow us to enter into the other person's perspective.

Further, Merton, Fiske and Kendall in Judd, et al. (1991) states that, for this type of interview (in depth) to occur, firstly, the interviews must be known to have been involved in the particular situation under investigations. In this research therefore, computers users will be interviewed because they are involved in the use of computers, they have been attacked by viruses before and they know how much other virus scanners has failed them. Secondly, the interview must be focused on the experiences of persons exposed to the situation in an effort to ascertain their perception of the situation. In the same line, in this research project, the interview will probe the experiences of computer users concerning their perceptions and challenges they receive from virus attacks. The researcher also played the role of interviewees because of his experience as a computer user. These steps will help the researcher to do a rigorous investigation in order to achieve the objectives for this research project.

It is of important to note that in-depth-interview does not follow a pre-planned list of questions. Sometimes the questions may develop spontaneously in the course of the interactions between the interviewer and the interviewee. Therefore, the researcher will enjoy the freedom to ask any questions that will need to be investigated (see the guided interview in Appendix A).

3.2.2. Observation or Field notes

Emersion, et al. (1995, p: 14) explain that, “Field researchers seek to get close to others in order to understand their way of life, to preserve and convey that closeness they must describe situation and events of interest in detail”. He further states that field notes “is to observe and record naturally occurring talk and interaction’. the researchers’ deeper concern lies in the actual, situated use of those terms in ordinary interaction (Emersion, et al. 1995, p: 140). Therefore, data was also collected using observation or field notes.

3.3. Population and Sample

3.3.1. Population

This research project focused on the computer users using different types of computers using windows, Linux, Macintosh and apple computers. These computer users were targeted because the computers they use have different virus scanners as a result they have different experiences of virus attacks which may influence their perceptions about virus scanners. Therefore, they are identified as key respondents who can supply the data required to accomplish this research project.

The criteria used in selecting these computer users in this project are the fact that; they are computer users, and they use computers with different operating system and type.

3.3.2. Sampling method

The sampling method used in this research project is the purposive one. According to Denzin and Lincon (2000, p: 370), “purposive sampling method seeks out groups setting and individuals where the processes being studied are most likely to occur”. Also, Silverman (2005) puts it that purposive sampling allows choosing a case because it illustrates some feature or process in which one is interested in. In this research project researcher purposively chose computer users of windows, Linux, Macintosh and Apple computers because they are the ones who can provide necessary data in order to make the once off virus scanner suitable for these computer users.

3.4. Data analysis Procedures

The researcher tried to collect data during working days that is from Monday to Friday (from February to August 2017). This was so because most of the respondents were readily available during these days. Firstly, the researcher had a chance to interview computer users from different fields using different types of computers. The interviewees were Lecturers from the computer science department and physics department at the University of Zambia as these had much more knowledge about computers and virus scanners. To collect balanced data the researcher also interviewed computer users from Zamtel Zambia in the IT section as they also had different perceptions of virus scanners. Secondly the researcher sent a sample of the interview questions to his supervisor for recommendations. Thirdly, the researcher gave the once off virus scanner to a number of computer users to try it and after trying it the respondents gave out their perceptions about the application. The data was later collected together ready for reporting.

3.5. Ethical Consideration

i) To stick to the stated guidelines such as confidentiality and honesty.
ii) To write or report as accurate as the original source/information (empirical data collected).

3.6. Methodology strengths and weaknesses

3.6.1. Strengths

Researcher was able to interact with computer users using different computers in all the fields. The environment was conducive for him to collect the data. Most of the respondents were able to explain the experiences or rather challenges they encounter with other virus scanners.

3.6.2. Weaknesses

It was difficult to collect the data as some of the respondents never understood the virus scanners they use and others do not even use any anti-virus or virus scanner at all. And most of the computers of the respondents never had visual basics which supports this version of the once off virus scanner.
The study was limited to 4 types of computer users that are those that use windows, Linux, Macintosh and Apple computers.

4. Results

4.1. Presentation of Empirical data

Presentation, interpreting and reporting of data are essential elements in an empirical research. Ghosh (1999) states that, "reporting of data is a critical examination of the collected data". It involves the verification of the problem for the study. Additionally, it involves the representation of the data, which can be done by tabulation, categorization, coding among other inferences. In this study, the researcher will attempt to report on the three modes of tools or instrument used to collect data. These tools include in-depth-interview, observation and literature review. The

above-mentioned tools were used to find ways and means of achieving the main of the study by releasing that they are valid and reliable.
The reporting of the results will be organized as follows;

- Personal details of respondents
- Relevance of once off virus scanner
- Computer users' perception on securing files and documents.
- Challenges faced in using of virus scanners

4.1.1. Findings on Personal details

All the respondents surveyed from the University of Zambia (UNZA), who are lecturers, hold PhDs in computer science and computational physics respectively. Also the respondents from Zamtel Lusaka, Zambia hold master's degrees in IT. The study reveals that the qualification of the respondents have a bearing on their perception of the creation of the once off virus scanner.

One the respondents from UNZA has been lecturing for less than 5 years, three have been in lecturing for years between 5 and 10 years while another one has been handling the computers for more than 10 years. All the respondents from Zamtel have been IT specialists for more than 6 years.

Since the majority of the respondents have been specialists in this field for more than 5 years, the researcher is confident that their long specialism experience will help to give balanced responses to the issues raised in this study.

4.1.2. Interview

4.1.2.1. Findings on the relevance of once off virus scanner.

When asked how relevant the once off virus scanner is to computer users, the majority of the respondents answered that it is very good as it needs no updates for it to scan effectively and efficiently. The respondents from Zamtel added on that the way it detects viruses is very much selectively which gives the user an opportunity to detect viruses before the files are stored in the computer. They further stated that Resource wise it demands less of an organization IT infrastructure or resources in the sense that it does not change how the system works or does not need you to alter how the system works for it to run. Simple to use and access in that it is a freeware.

Another question was asked on some of the measures that should be taken to improve virus detection, majority respondents from both UNZA and Zamtel said that virus scanners should be user friendly because not all computer users are that computer literate. They gave an example of smadDAV which is an USB virus scanner they said it scans automatically which makes it easy to use by any computer user.

4.1.2.2. Findings on Perceptions on Benefits of having once off virus scanner

When the lecturers and IT specialists were interviewed on the importance of the virus scanner and virus scanning, the following were the actual feed backs;

Majority of the respondents said that a computer without virus scanner usually; run slowly, boots up, its parts of the system are damaged, viruses amends its operating system without the user's knowledge, can be used to steal personal details, its viruses attacks other machines on a network and sends out unauthorized messages.

They further said that just a human virus a computer virus can be dangerous and possibly infectious to other computers. They can be spread between computers, and can also be transported via email. Respondents added on that it is not limited to software however, as they can also be transported via USB drives and other types of media. So they said it is very important to have a once virus off virus scanner that detects viruses in any file before it is stored in one's computer.

Zamtel respondents said that antivirus is the "policeman" at the gate of a computer system. It protects the computer from incoming threats and seeks out, destroys and warns of possible threats to the system. New viruses are coming out all the time. It is the job of the antivirus software to keep up with the latest threats. This is achieved by daily updates of the antivirus databases definitions, which counteract the latest threats to provide constant protection. Now having a virus scanner that does not need daily updates for effective protection it is a much important application to have a computer.

4.1.2.3. Findings Regarding Challenges faced when using virus scanners

The respondents were asked to state the challenges they faced as they used virus scanners as compared to the once of tool,
It was found that with the once of tool it was not only easier for the interviewees to use but also to learn to use it the fact that it is able to be tailored to a specific user's needs gave it the most advantage thus creating challenges even for people who would want to manipulate the information because they would not be able to know how each individual has been using the tool

These findings helped the researcher enable and make sure that the tool works to its desired specification and design

Discussion

The findings of this research lead the researcher to focus more on a robust but yet simple to understand project because of the wide range of people that would deal with the tool but yet again their comes in the aspect of security can it be too simple and compromise on security? NO. Hence the tool was designed to be easily manipulated to create a wide frame in which if and attack was to happen it would take a very long time for the hacker to manipulate just one file

This leaves us with two questions

1 is the tool UN hackable?

2 is the tool to simple?

These questions can only be answered after the final part of testing is done with is consumer usage.

Conclusion

In the concluding remarks the small brief report the researcher has tried to explain how the tool works and what it will do he explained all the added advantage it has over other tools it is a really unique tool that will add security to the computer field and it will in future be the tool that everyone will want to have the once off virus scanner is as shown one of a kind.

Recommendation

After having carried out the research the researcher recommends

1. Always having files you want to be protected from specific virus scanned

2. Since it’s an open source software, others can input better codes to help the tool perform better and be used in other fields e.g. processing and data analysis

3. The researcher also advises users to exercise caution when operating this tool for maximum and accurate results

Acknowledgements

I cannot express enough thanks to my course mates for their continued support and Encouragement: Dr. RICHARD SILUMBE, my adviser and INNOCENT NSUNGA my lecturer. I offer my sincere appreciation for the learning opportunities provided by the school (information Communication University).

My completion of this project could not have been accomplished without the support of

My classmates, MR THOLE, ASSAN and MR OBED; and relatives. Thank you for allowing me time away from you to research and write. You deserve a trip to Disney! Thanks to my parents as well, Mr. and Mrs. CHAILA. The countless times you kept pushing and guiding me through the hectic schedules, I will not forget.

REFERENCES

[1] M. bishop. An overview of Computer Viruses in a Research Environment. Obtained from ftp://ftp.informatik.unihamburg.dde/pub,virus,texts/viruses/malogic.zip

[2] V. Bontchev. Are good computer viruses still a bad idea? Obtained from ftp://ftp.informatik.unihamburg.dde/pub,virus,texts/viruses/goodvir.ziop

[3] V. Bontchev. Possible virus attacks Against Integrity Programs and how to prevent them. In proc. Second International virus 1992. Obtained from ftp://ftp.informatik.unihamburg.dde/pub,virus,texts/viruses/attacks.zip

[4] F. B. Cohen. A Short In Computer Viruses, JohnWilley and Sons, Inc, Second eddtion 1994.

[5] M. W. Eichin and J. A. Rochlis. With Microscope and Tweezers: An On Computer Virus of November 1998. Obtained from ftp://atheenadist.mit.edu:21/pub/virus/mit.PS.

[6] H. J. highland. A macro virus. Computer & Security, may 1989, pp.178-182.

[7] T. A. Longstaff and E. E. Schultz. Beyond prelinary analysis of the ANK and OILZ worms: a case study of malicious code. Computer & Security, Volume 12, Number 1, pages 61-77, 1993.

[8] S. Magruder. High-level language computer viruses – a new threat? Computers & security, volume 13, number 3, pages 263-269, 1994

[9] Y. Radai. Integrity Checking for Anti-Viral Purposes Theory and Practice. December 1994. Obtained from http://shum.cc.hugi.ac.il/radai/virus.htm; an improved version of "Checksumming Techniques For Anti-Viral Bulletin Conference, September 1991.

[10] D. Seeley. A Tour of the Worm. Obtained from http://www.ja.net/newfiles/janinfo/cert/seeley/tour.ps

[11] E. H. Sparford. The Internet Worm Program: An Analysis. Purdue Technical Report CSD-TR-823. Obtained from http://www.ja.net/newfiles/janinfo/cert/stafford/Internet_worm.ps

[12] E. H Stafford. The Internet worm Incident. Purdue Technical Report CSD-TE-933. Obtained from
http://www.ja.net/newfiles/janinfo/cert/stafford/IWorm2.ps

[13] S. R. White. Cover Distributed Processing with Computer Viruses. In Advances in Cryptology – Crypto’89 proceedings, pages 616-619, Springer-Verlag, 1990

[14] A. young. Cryptovirlogy: Extortion-Based Security Threat and Countermeasures. Inproceedings og 1996 IEEE symposium on Security and Privacy. Obtained from
http://ww.cs.collumbia.edu:80/~ayoumg/-iee96.ps.gz

[15] U. S. Department Of Energy’s Computer Incident Advisory Capability Macro Viruses. Obtained from
http:// ciac.llnl.gov/ciac/bulletins/g-10a.shtml

[16] Computer Viruses Catalog (Macintosh Viruses). Obtained from
ftp://ftp.informatik.unihamburg.dde/pub,virus,texts/catalog/macvir.zip

[17] Computer Virus Catalog (PC/MSDOS Viruses). Obtained from
ftp://ftp.informatik.unihamburg.dde/pub,virus,texts/catalog/msdos.zip

[18] Various Contributors. Virus L/com.virus FAQ. Obtained from
http://www.bocklabs.wisc.edu/~janda/virl_faq.html.

[19] Virus Contributors. Atl.comp.virus FAQ. Obtained from
http://www.bocklabs.wisc.edu/~janda/virl_faq.html.

[20] Postin by Graham Cluley on 23 Jul 1996 in the USENET newsgroup atl.comp.virus.

[21] Posting by Brian Seborg on 15 august 1994 in the USENET newsgroup. comp.virus. VIRUS-L Digest, Volume7, Number 68. Obtained
http://csrc.ncsl.nist.gov/virus/virrul*)*

[22] Virus Privalence Table compiled using reports sent to and collected by Virus bulletin.
Found at *http://ww.virusbtn.com.prevalence/*