Testing Times for Trojans

Ian Whalley

IBM TJ Watson Research Center, PO Box 704, Yorktown Heights, NY 10598, USA
Tel +1-914-784-7808 • Fax +1-914-784-6054 • Email inw@watson.ibm.com

1 Abstract

In the field of computing, Trojan horses have been around for even longer than computer viruses – but traditionally have been less of a cause for concern amongst the community of PC users. In recent years, however, they have been the focus of increased attention from anti-virus companies and heightened levels of user concern.

This paper aims to investigate the Trojan phenomenon; particular attention will be paid to the claims made in the field of NVM detection and those made by those who aim to test the vendors' claims.

In addition, various attempts to define Trojan horses will be evaluated, and a new definition will be suggested. It is not expected that any such definitions will solve any of the basic problems, but they will help shed light on certain aspects.

The author will also investigate further techniques to test the claims of software vendors in this very difficult area.

2 Introduction

As the 1990s draw to a close, the anti-virus industry is undergoing a sea change. Those who have followed the AV industry for any length of time are probably bored of hearing such sweeping generalisations, but this author believes his statement to be correct, and will attempt to justify it.

The risks to an average PC in 1999 are very different from the risks to that same PC in the early- and mid-90s, and the manufacturers of traditional anti-virus products are finding that they have to change their products in order to deal with these new threats. However, in order for the transition to be successful, other parts of the anti-virus industry will need to change along with the products.

The most obvious trend at the time of writing is in the area of network-aware viruses. In recent months we have seen more and more occurrences of this type of malware, and this threat looks set to grow as we enter the next century. The problem of malware that spreads using both the Internet and local-area networks is all too evident, and the art of defending against such malware is urgently in need of a boost.

There is concern, however, over another threat – the Trojans of the title.

3 Trojan horses

It may initially seem somewhat odd that this paper sets out to discuss Trojan horses – Trojans have been around for a while, after all, and surely nothing much has changed. Indeed, the fact of the matter is that the average computer user is not encountering Trojan horses with any regularity (or indeed at all!) – the reader is referred to [1] and [2] (updated in [3]) for figures and a cogent analysis of the overall Trojan situation.

The standard example of an environment in which the threat of Trojan horses is very real and prevalent is America On-Line (*AOL*). There are several reasons why *AOL* is an ideal environment for Trojan horses – including:

- **Customized features**
  In order to be the 'easiest and most convenient Internet online service' [*sic*] referred to in its marketing materials [5], *AOL* installs custom software on the user's PC. This software manages access to the *AOL* user forums, and deals with email and instant message and the like.
- **Large user population**
  AOL itself claims to have over 17 million users [6], and claims to be the largest such service in the world.
- **Unskilled user population**
  AOL provides for its users a consistent and widely varied electronic environment in which the users can interact with each other. In this, it is not unique – however, it is unique in that it makes it extremely easy for the users to use the service; the *AOL* client software is wonderfully easy to install and configure. In addition, *AOL* has achieved a position of market dominance in the area of providing online services for people who don't know anything about computers.
- **Anonymity**
  When you create your account with *AOL*, it is necessary to choose a so-called 'screen name' – this is the name by which you will be known.[1] In the parlance of the on-line privacy community, a screen name can be compared to a 'nym'[2], and *AOL* itself can be compared to a so-called 'pseudo-anonymous remailer'.[3] Without the assistance of the *AOL* authorities, there is no way to trace the screen name 'Alice1701' to the owner of that account.

If these facts are combined, the result is an environment in which the majority (those who know very little) are wide open to abuse from the minority (those who are experts[4] with computers in general and *AOL* in particular).

Not surprisingly, such abuse has been taking place, with varying degrees of success, over the last few years, and looks set to continue into the future. The techniques will be entirely laughable to most people reading this paper, but that doesn't stop them being successful and worthwhile to the attackers. The precise techniques are not really important, and there are many variations on a theme. A typical scenario would be as follows:

- **Technology**
  Alice (the attacker) writes a program that, when run on a given computer, determines the *AOL* account information held on that computer, and communicates that information to Alice.
- **Social Engineering**
  Alice then sends this program to Bob (the victim). She writes a message to go with the program claiming that she is an *AOL* technician, and Bob should immediately run this program due to a security problem, to speed up *AOL* access 400%, to diagnose a communications problem, for any one of a long list of semi-technical sounding reasons.

---

[1] The large number of *AOL* users accounts for the prevalence of names such as 'Alice1701' – when asked for a name, most users will optimistically choose something like 'Alice'. When (inevitably) that name is not available, the *AOL* software suggests a version of the name you asked for with numbers appended.

[2] 'Nym' is a contraction of 'pseudonym', which is defined as 'A false or fictitious name, esp. one assumed by an author' [7].

[3] 'A remailer is computer service that privatizes [*sic*] your e-mail … Traditionally, a remailer allowed you to send electronic mail to a Usenet news group or to a person without the recipient knowing your true name or your e-mail address … A pseudo-anonymous remailer [is one where] the operator [knows] your real email address. Your privacy is only as good as the remailer operator's power and integrity to protect your records' [8].

[4] The definition of 'expert' in this context is the traditional 'someone who knows 10% more about something than 90% of the population'.

- **Execution**
  If Bob runs Alice's program, his account information is communicated in some way to Alice. Alice can then use Bob's *AOL* account, just as if she was Bob. If Bob doesn't run the program, Alice hasn't lost anything, as Bob is fairly unlikely to report Alice to the *AOL* authorities, and Alice is probably using a hijacked account anyway. As Alice sent the exploit to multiple victims, the chances are that some of them will run it, and Alice can move on to using a different account.

The phenomenon of the so-called '*AOL* Trojan' has been well documented in the past, and the reader is referred in particular to [4] for more information on the subject.

4 So what?

It is a sad fact that this type of attack is very easy to carry out. Thanks to modern programming languages and environments, writing a program of the type outlined above is well within the reach of many people. This trend – of bringing programming to the masses – is of course to be applauded; however, it is a mixed blessing. For example, if more people knew how to break into the average family car, there would be a higher rate of car crime. If more people knew how to commit credit card fraud in near untraceable ways, there would be a higher rate of credit card fraud. In the same way, as more people discover how easy it is to produce this type of Trojan, more and more people will start to produce it, and the problem will increase.

5 How is this relevant?

Let us suppose that, in the example above, Bob is running the top of the line anti-virus product, *McNortSophOlomon Anti-Virus* (*MNSOAV*). Bob has kept the signatures in his copy of *MNSOAV* up-to-date – once a week it updates itself automatically from the Internet. From Bob's point of view, he should have been protected – with the single (albeit significant) exception of running mysterious software sent to him from an untrusted source. However, Bob could argue that the precise reason he pays his subscription fees to the *McNortSophOlomon Corporation* is that he wishes to be protected if and when he forgets himself and does something silly. After all, when Bob is buying a car, he looks for one with airbags – not because he intends to go right ahead and drive his car at high speed into a brick wall, but so if and when an unfortunate accident does occur, he will be protected.

However, *McNortSophOlomon Corporation* could argue that their product is, after all, called *MNSO* ***Anti-Virus***, and Bob is therefore not entitled to assume that he is protected against Trojans. However, upon learning this, Bob can promptly cancel his subscription to *MNSOAV* (which, after all, isn't helping him and doesn't do what he wants), and subscribe instead to a competing product, *Dr InocuTonAfeeOs Anti-Virus* (*DITAOAV*), which does advertise itself as detecting Trojans.

Not only does *DITAOAV* advertise itself as detecting such things, Bob has done his research. *DITAOAV* has performed admirably in some recent tests examining the ability of this type of product to handle Trojan horses, and Bob feels reassured.

6 The tests

What of the tests in which Bob has put his trust? Consumer tests are essential in the modern world, due to the incredible complexity of the products available – after all, when Bob buys his car with airbags, he is unlikely to do his own consumer testing, unless he has a considerable amount of skill, time, and money (and luck…). Of course, matters are not this clear-cut in the AV arena – because Bob is clearly aware of viruses and the like, it is likely that he has kept a copy of any viruses he has encountered over time. This is standard practice, and whilst AV companies have attempted to discourage it in the past, it is a reasonable base-line test that Bob can use to make some trivial decisions about his

anti-virus product.[5] However, Bob is a knowledgeable chap, and realises that the tests he can do on his own are not enough. Therefore, he peruses the results of third-party tests, and uses their results to help him decide with products he should consider and which he should not (for more information, the reader is referred to [9] & [12]).

Some time ago, testing agencies realised that it would be necessary for them to introduce tests to determine the ability of anti-virus products to handle Trojans, as two things were becoming increasingly evident:

1. Users (customers) wanted to know whether or not the products that they were using and considering using would protect them against the various types of Trojans out there.
2. Producers (anti-virus companies) wanted to be able to demonstrate that their products could indeed handle such things.

These two reasons are nothing new – they are the same reasons that have justified traditional anti-virus product testing for many years; there is a demand for some form of comparison testing of the available products, and so a supply of such tests inevitably results.

It is interesting to pause for a moment and ponder – given the research documented in [1], why are manufacturers and users even remotely interested in Trojans? It appears to be another case of the 'positive feedback effect'. That is to say, as soon as one producer takes the leap and advertises its products as capable of defending against Trojans, other producers have to jump into the game as well. Both the producers and the users then demand (for different reasons) testing of these 'new' features. Producers want some way to differentiate product from product, and tests can provide this; users want to know if the product they are considering buying is capable of doing that which it claims to do. The reader is referred to [9], [10] & [11] for more information on the symbiotic relationship between producers and reviewers of anti-virus products, and other related topics.

7 Current state of the art in anti-Trojan testing

As demonstrated above, the marketplace has decided (in the parlance of capitalism) that anti-Trojan testing is required. Given this fact, an examination of the current tests involving Trojans is in order. The following sections cover the four major professional testing bodies.

7.1 University of Hamburg Virus Test Centre

The Virus Test Centre (VTC) is an academic research institution that publishes reviews of anti-virus software at approximately six-monthly intervals [15]. VTC does test the ability of anti-virus software to detect Trojans – see [16]. In particular, note the following:

> *As VTC malware tests demonstrated that almost all AV products are able to detect a significant part of malware, VTC now considers it´s* [sic] *malware test as *mandatory part of VTC tests*. Indeed, all essential AntiVirus manufacturers agreed that their product also be tested against VTC´s malware testbeds (one enterprise did not* [sic] *agree and was dropped from the list of products to be included in VTC tests).* [17]

Unfortunately, VTC could not be contacted in time this paper to be submitted.

[5] This author will continue to refer to 'anti-virus products', in spite of the fact that in the area under discussion, these products are not being used in a specifically anti-virus capacity. This blurring is intentional, and mimics the blurring that is evident in the marketplace.

7.2 West Coast Labs

West Coast Labs is owned by West Coast Group (a privately held company located in Swansea, UK), which also owns the security publication 'Secure Computing' [19]. West Coast Labs is responsible for the Checkmark scheme for certifying security-related products – the anti-virus Checkmark scheme of the time is discussed in [9], although changes have been made since then [18].

West Coast Labs have recently introduced a 'Trojan Checkmark', part of the description of which is as follows:

> *For a product to be certified to Trojan Checkmark, the product must be able to detect all trojans in the West Coast Labs Trojan test suite. The product should not cause any false alarms (based on testing against the West Coast Labs false alarm test suite).* …
>
> *A trojan test suite is maintained by West Coast Labs. The list of trojans will be published by West Coast Labs (possibly on the Checkmark web site) and copies of Trojans will be provided at the discretion of West Coast Labs to* bona fide *solution developers and members of the Trojan Checkmark scheme.* …
>
> *Note*
> *(a) The Trojan Checkmark may be subject to change as a result of developments in the field of trojan threats and testing.*
> *(b) Additional levels of certification will be introduced as appropriate and in due course.*
> *(c) Developers agree, as a condition of registering for the Trojan Checkmark, to provide copies of their collections of trojans to West Coast Labs.* [20]

Note carefully the marked sections. Upon signing up for the Checkmark scheme, vendors provide the Trojans which they have in their collection to West Coast Labs. In addition, producers in the scheme can receive copies of at least part of the West Coast Labs test set.

West Coast states that it found maintaining the list of Trojans mentioned in [20] to be 'totally impractical, as they came in faster than anticipated, so this idea had to be abandoned' [21]. In addition:

> *I believe that all of our Trojan samples have been supplied by AV companies rather than drawn from the Wild. First we try to match them to existing samples; if they aren't already represented then we confirm that they are Trojans rather than viruses or innocent programs by examining their functions, either by analysis or execution (we find names reported by AV products are often a good indication of the file's functions). Borderline cases are omitted (as they are elsewhere in our collection – for instance, we decided not to include in our tests ALREADY.COM)*
>
> *Apart from an occasion when Trojans were damaged in the process of sending them to the customer, our samples haven't been questioned.* [21]

7.3 Virus Bulletin

Virus Bulletin [22] is a privately held company located in Abingdon, UK, and is under the same ownership as anti-virus company Sophos. At the time of writing, Virus Bulletin does not perform any type of anti-Trojan testing [23].

7.4 ICSA

The International Computer Security Association (ICSA) [24] is a privately held company located in Virginia, USA. At the time of writing, the ICSA does not perform any type of anti-Trojan testing [25], however, it is involved in the creation of a group to examine the problem of defining what is and what is not a Trojan – for more information, see section 10.

8 Theoretical test requirements

It is useful to consider the similarities between requirements for building a good set of viruses – as will be seen, the requirements for constructing a set of Trojans are not as different as might have been thought.

When building a set of viruses against which to test anti-virus products, the tester is required to verify the viral nature of each sample in his set. This is a worthwhile exercise for the tester for reasons other than technical exactitude – if vendors cast aspersions on his results, the tester can be certain, and can demonstrate, that his samples are valid. This most basic requirement of test-set construction has been documented in several works – refer to [13], [26], and the works they reference.

This requirement is just as fundamental to the art of anti-Trojans product testing – without 'proof' that the samples that the tester is using genuinely represent Trojans, the tester has no hope of performing tests which are either reputable or repeatable. Therein lies a fundamental problem.

8.1 The verification problem

How can the tester prove that the samples he has in his test-set are valid samples of Trojans? If the tester were doing tests against samples of *viruses*, such verification would be fairly simple, if time-consuming. The tester would simply take samples from the test-set, and replicate them. If the samples could be replicated, then the matter is closed – the samples are viruses. Of course, there are other issues – such as are they samples of the correct virus (particularly relevant for ItW [14] tests) – but these are secondary in this discussion to the fundamental question of 'Is it a virus?'

In the case of viruses, the sample verification process can be rolled into the initial sample creation process – when the tester makes his initial samples, he can produce additional generations of replicants in order to verify that the samples he will use for his test-set are capable of replicating correctly. The tester will wish to document the fact that he has done this, and also document the system upon which the virus was replicated – this information will save time later on when and if he needs to proof the validity of the samples in the set.

This system only works thanks to the very basic definition of 'virus' – which for the purposes of this paper will be stated as 'code which can make copies of itself' [28]. Provided the tester can prove that his samples fulfil this requirement, he is on reasonably firm ground.

However, Trojan horses do not have this requirement – and this leaves both testers and producers with a remarkably basic question.

9 What *is* a Trojan horse?

There follows (in no particular order) some previous definitions of the phrase 'Trojan horse':

> *A Trojan horse is a program which performs (or claims to perform) something useful, while in* [sic] *the same time intentionally performs, unknowingly to the user, some kind of destructive function. This destructive function is usually called a payload.* [26]
>
> *A Trojan horse is a program which performs functions other than those stated in its specifications. These functions can be (and often are) malicious.* [27]
>
> *A Trojan horse is, as the name suggests, a program which is allowed onto the user's PC under false pretences, whereupon it has undesirable side effects.* [28]
>
> *Trojan horse: A computer program with an apparently or actually useful function that contains additional (hidden) functions that surreptitiously exploit the legitimate authorizations of the invoking process to the detriment of security.* [29]

> *A program which someone tells you is legitimate software, but which actually does something other than what the person claims it will do.* [2] (Emphasis preserved from original)

> *A program which the user thinks or believes will do one thing, and which does that thing, but which also does something additional which the user would not approve of.* [3]

These definitions are from various eras of computing – from the mid-80s to the present day, and in some senses reflect the changing nature of the problem. Almost inevitably, some are more useful and appropriate than others – of particular relevance, in the opinion of this author, is the marked definition. However, a suggested new version is:

> *A program which the user thinks or believes will do one thing (the 'perceived purpose'), and which may or may not do that thing, but which also does something else which is not necessary to accomplish the perceived purpose, and of which the user would not approve (the 'payload').*

The reader will notice that this definition is entirely subjective in nature – however, this is also true of the definitions suggested by other authors listed above. The main difference is that many of the other definitions do not make their subjectivity clear – the author's definition (and the definition from which it is derived) make that subjectivity explicit.

It seems clear to this author that it is not possible to produce a definition of Trojan that is not subjective in nature. Bearing this in mind, the advantages and disadvantages of the above definition are discussed below.

9.1 Advantages of the suggested definition

9.1.1 No mention on 'intent'

Many of the current definitions of Trojan refer to the program 'intentionally' doing something bad to the host system. In the opinion of this author, if Bob executes a program which is clearly described in the accompanying documentation as a word processor, but which formats some of the tracks on his hard drive (or damages his Linux installation, or some other 'bad thing'), intent is not an issue. That program is a Trojan whether or not the author wrote those the destructive routines deliberately (and with malice aforethought.)[6]

9.1.2 No requirement to carry out perceived purpose

To use the earlier example, when Bob receives a program from Alice that is described as a patch for the AOL client to speed up access 400%, that becomes its perceived purpose. If that program actually does something bad to Bob's computer, or transmits his account information to Alice, or anything along those lines, that program is a Trojan even if it does not also carry out the perceived purpose.

9.1.3 No requirement for the payload to be destructive

The inclusion of words like 'destructive' and 'damaging' in some definitions needless restrict those definitions. If Alice's program transmits Bob's account information to Alice, that is without doubt an action of which Bob would not approve. However, it is not in and of itself destructive – it may become indirectly destructive (in some senses), depending on what Alice does with the information.

---

[6] Admittedly, it is hard to imagine a word processor accidentally formatting bits of a hard disk, but the point is valid nonetheless.

9.1.4 No mention of specification

The word ‘specification’ has many connotations in the field of computer science, none of which apply particularly well to the field of the user’s expectations of what a given piece of software is going to do. Talking instead of the perceived purpose of that piece of software more clearly illustrates the situation, whilst leaving the precise definition very open.

The perceived purpose is something that will vary from person to person – this is absolutely essential to a modern definition of a Trojan. Not only that, but it will vary according to the way in which a given program is distributed. For more discussion of this point, see section 9.3.

9.2 Disadvantages of the suggested definition

9.2.1 No mention of specification

One of the advantages mentioned above (section 9.1.4) is also a disadvantage – the fact that whether or not something is a Trojan can vary from person to person, and according to the way the object is presented, is a real problem for those producers attempting to defend against them. Again, for more discussion of this point, see section 9.3.

9.2.2 No mention of ‘intent’ (or: Bugs are Trojans?)

A side effect of the fact that there is no requirement for intent is the fact that, under some circumstances, bugs could be considered to be Trojans. As mentioned above (section 9.1.1), if the bug causes behaviour ‘of which the user would not approve’, then this is entirely correct. However, if the bug causes a trap, or causes non-fatal errors (for example, it causes Word to make irrational and apparently random claims of incorrect grammar), it could be argued that this is behaviour ‘of which the user would not approve’, and yet it is surely unreasonable to classify the program as a Trojan simply because of insufficient sanity checking or a poor grammar checker.

This is due to the entirely subjective nature of the definition. However, within the boundaries of the definition, it is valid – some bugs (the accidental destruction of data on the disk mentioned above, for example) would indeed cause many people to classify the program exhibiting them as a Trojan.

9.3 The problem of perceived purpose

From the point of view of both producers and testers, introducing the concept of a program’s perceived purpose to the definition is a real problem, for a number of reasons, a couple of which are briefly mentioned below – the reader is no doubt able to think of several others.

9.3.1 Renaming

As briefly mentioned above, the perceived purpose of a given program can vary according to how that program is presented. This is true right down to that most basic of operations, renaming an executable. In many cases (particularly on the Internet), the filename is the only indication which a user has pertaining to the supposed purpose of the program.

For example, consider the case of the malware Happy99 (also known as Ska). This virus attacked Windows 9x TCP/IP services, and was able ‘watch’ what the user was doing, and then send itself via email and Usenet postings to the same people and groups to which the user was posting. Messages sent by Happy99 arrived with no supporting text, merely an included executable file called HAPPY99.EXE [30]. The recipient of such a message has no other indication of the nature of the attachment.

9.3.2 Other supporting documentation

Even if producers were able to account in some way for the name of a program, that is not enough. The program may come with other documentation (in the form of README.TXT, help files, even printed manuals), all of which can affect the perceived purpose of the program itself.

Now, the situation seems impossible – and it may very well be. The perceived purpose of a program will almost inevitably change depending whether or not the user actually reads the documentation! What if the documentation is in a language that the user does not speak? What if the documentation has been mistranslated into the user's native language, and the meaning has been damaged?

10 In reality

The fact that simply looking at the bytes which make up a given object is no longer enough to determine whether or not it should be flagged as a risk has far-reaching consequences. In the opinion of this author (and fortunately for users), the scenarios described above are comparatively unlikely to occur in significant numbers in the user community. However, this does not alter the importance of at least attempting to resolve some of these issues in the world of anti-Trojan testing.

To return to the more general problem of determining the perceived purpose of a given program, one of the testing agencies mentioned above is in the course of setting up an interesting-looking project to attempt to define the scope of the problem. Some of the plans for this project are outlined below.

10.1 ICSA's Malware think-tank

As an attempt to move some distance towards resolving the problems discussed above, the ICSA is setting up a group of people to help construct a Trojan test-set. This is in response to requests from corporate customers of the ICSA, who perceive a need for such testing (given the research in [1], [2] & [3], presumably this is at least in part to the factors outlined in section 6).

Current plans are that the group will be overseen by a board drawn made up of members from the ICSA, the corporate world, academia, and AV producers. Underneath the board comes a larger number of people from each of the above categories – between them, they will attempt to decide whether or not something is malware.

Each member of the group will examine submissions in an attempt to determine whether or not a given program is malware. If they look at a given program, each member can (fundamentally) vote one of three ways – 'definitely malware', 'definitely not malware', or 'can't tell'. If the group is split, there are weighting factors applied to each possible vote, and the combination of those weighted votes will place the suspect program in one of the three possible categories.

In essence, the ICSA is proposing trial by jury for suspected malware – a system which should logically be one of the best available, given the entirely subjective nature of the definition of Trojan. A group of people, skilled in the art, must reasonably conclude that the suspect program is a Trojan before it is declared guilty.

Whilst this is, on the face of it, a promising idea, there are several factors that could cause it difficulties:

- **Time**
  Most modern Trojans are compiled high-level language programs, and analysing such things is a time-consuming process – the members of this group are not working for it full-time, and so even initially, there may be time problems.

  Of course, using the definition suggested above (section 9), reverse engineering is not necessary – the group members need simply execute the suspect program, and see if its behaviour makes it a Trojan. This does not mean that the group members simply need to execute everything that is submitted to them without any thought – the actions of the program need to be monitored and analysed (presumably by hand), and the group member will have to attempt to create a suitable environment for the suspect program. Executing a suspected AOL password stealer on a machine without the AOL software installed may well result in the group member submitting a 'definitely not Trojan' vote.
- **Volume**
  Even if the group has enough resources initially, if the volume of suspected malware grows over time, resources will become increasingly stretched.
- **Skill**
  The number of people with sufficient skill and training to make the decisions required by such a project is not all that great – there is a danger of manpower shortages.

However, this author does not wish to predict doom for such a group before it has got off the ground – the results of an appropriately even-handed set of deliberations such as are the apparent intention of the group will undoubtedly make for an interest research topic in and of themselves, and could shed considerable light on this murky area.

As an example, it would be educational to give the group copies of the following pieces of software, and ask them to determine whether or not they are malware:

- NetBus 2 [31]
- BackOrifice 2000 [32]
- Microsoft SMS [33]

It would also be instructive to determine whether or not the opinions of the group's members on each of the above programs were altered depending upon whether or not they were aware of the controversy surrounding these programs. The reader is referred to [34], [35] & [36], in addition to numerous threads on the alt.comp.virus newsgroup.

11 Return to Testing

What, then, of anti-Trojan testing? As discussed previously (see section 8), the requirements for anti-Trojan testing are harder to define than those for anti-virus testing. However, this author suggests the following process for building a Trojan test-set.

- **Acquisition**
  Samples are acquired from any of the usual legitimate sources – be that a vendor collection, real victims of the Trojan, etc.
- **Verification**
  Each sample is examined, and a decision is made whether or not it is a Trojan. This is the same requirement that exists for viruses, however (as discussed) the determination is less clear-cut in the case of non-viral malware.
- **Documentation**
  Documentation is prepared for each sample that describes what it does, why it is judged to be malware, and how that determination was made. This documentation will probably include a CRC of the sample, to ensure that the documentation remains connected with the correct sample. This requirement is not unduly onerous – the documentation for many samples will be extremely brief.

- **Deployment**
  Each sample is then added to the tester's set. The documentation associated with that sample may, if the tester wishes, be made available to producers whose products are tested against the set. This author recommends that this be done, in order to clarify the test requirements (the tester may, if he wishes, remove the CRC, or not document the algorithm used to generate the CRC, to prevent manufacturers using the CRC in their identification). In addition, the documentation should be made available to the consumers of the test results.
- **Testing**
  Products are then tested against the set. The tests-set should be under version control (numerous commercial and free open-source products are available that can assist in this) – this allows the tester to retrieve the test-set used in any given test (and the documentation associated with that test-set) without difficulty at any given point in the future. This is invaluable if and when producers question the results. In addition, the precise binaries (certainly the distributed binaries, possibly also the binaries as installed on the test machines) are placed in version control, so that they too can be retrieved in the future.
- **Modification**
  It may well be that, upon discussions with both the producers of the products under test, and the consumers, that the tester decides that some of the malware that he placed in the test-set should not, in fact, have been there – perhaps he has classified a sample as malware, when in fact it is not. In this case, the tester will need to modify the set – this is not something of which the tester should be overly ashamed, but obviously he should attempt to ensure that the same type of problem does not happen for the same reasons again! The tester modifies the current version of the set, and documents this modification in a document describing its overall makeup and history. This document is also made available at the time of the next test.

By the process described above, the tester can ensure that his test-set is both maintainable and justifiable. However, the reader may notice that the list above glosses over the tricky matter of 'verification' – the process by which the tester decides whether or not a sample is malware. Certainly it is not suitable for the tester simply to take samples given to him and arbitrarily put them in the test-set without any attempt at verification – this will very quickly lead to pollution of his set, and possibly even manipulation of his tests by unscrupulous producers.

It is to be hoped that the ICSA's approach (see section 10.1) will assist in this area – the author and readers must wait and see.

12 Conclusion

Given the scenarios outlined in the early part of this paper, the continuing interest in anti-Trojan testing seems doomed to continue, regardless of whether or not the average user is actually at any sort of risk from Trojans.

In the field of testing, two things appear certain:

- The current standard of anti-Trojan testing can be improved to a certain extent by careful justification and documentation of samples and test-sets, as outlined in section 11.
- Even with such precautions, the contents of the test-sets will always be a matter for controversy. The subjectivity of the all definitions of Trojan (section 9) will inevitably lead to disputes concerning whether not certain files are appropriate for inclusion in a test-set. Faced with such disputes, producers will have to decide whether they wish to include identification for certain samples, even if they do not believe them to be Trojans, or to get lower detection scores in reviews.

It remains to be seen whether the relevance of such tests will increase in the future – due either to the testers raising their standards, or to the Trojan problem increasing.

13 References

[1] Sarah Gordon and David M Chess, *Where there's smoke there's mirrors: The truth about Trojan horses on the Internet*, Proceedings of the Eighth International Virus Bulletin Conference, pp 183-204, October 1998, <http://www.av.ibm.com/InsideTheLab/Bookshelf/ScientificPapers/Gordon/Trojan/Trojan.html>.

[2] Sarah Gordon and David M Chess, *Attitude Adjustment: Trojans and Malware on the Internet*, Proceedings of the EICAR Conference, February/March 1999.

[3] Sarah Gordon and David M Chess, *Attitude Adjustment: Trojans and Malware on the Internet (an update)*, Preprint, 1999.[7]

[4] Igor Muttik, *Trojans – new threat*, Proceedings of the International Virus Prevention Conference – Protecting the Workplace of the Future, April 1998.

[5] *About AOL.COM*, <http://www.aol.com/info/about.html>.

[6] *AOL Profile*, <http://www.aol.com/corp/profile/>.

[7] *Oxford English Dictionary*, Oxford University Press, 1989.

[8] André Bacard, *Anonymous Remailer FAQ*, <http://www.well.com/user/abacard/remail.html>.

[9] Ian Whalley and Richard Ford, *Testing the Untestable: the Hidden Roadblocks to Anti-Virus Testing*, Proceedings of the Eighth International Virus Bulletin Conference, pp 1-13, October 1998.

[10] Sarah Gordon and Richard Ford, *Real World Anti-Virus Product Reviews and Evaluations – the Current State of Affairs*, Proceedings of the Nineteenth National Information Systems Security Conference, Volume 2, pp 526-538, 1996.

[11] Sarah Gordon, *Evaluating the Evaluators*, Virus News International.

[12] Ian Whalley, *Re: Categorizing/Rating Anti-Virus software*, alt.comp.virus, 28 July 1999, Message ID: <slrn7pv4j3.1dk.ian@cromarty.whalley.org>, available at <http://www.deja.com/[ST_rn=ps]/getdoc.xp?AN=506377012&fmt=text>.

[13] Vesselin Bontchev, *Analysis and Maintenance of a Clean Virus Library*, Proceedings of the Third International Virus Bulletin Conference, September 1993, pp 78-89, <http://www.virusbtn.com/OtherPapers/VirLib/virlib.txt>.

[14] The WildList Organization International, <http://www.wildlist.org/>.

[15] Virus Test Centre, <http://agn-www.informatik.uni-hamburg.de/vtc/eng1.htm>.

[16] 'Scannertest March 1999', <ftp://agn-www.informatik.uni-hamburg.de/pub/texts/tests/pc-av/1999-03>

[17] 'Background, Aims of this test', <ftp://agn-www.informatik.uni-hamburg.de/pub/texts/tests/pc-av/1999-03/2PROLOG.TXT>

[18] West Coast Labs, <http://www.check-mark.com/>

[19] *About West Coast Labs*, <http://www.check-mark.com/checkmark/general/about.htm>.

[20] *West Coast Labs Developer's Briefing No. 45, Trojan Checkmark Level One*, <http://www.check-mark.com/checkmark/brief-45uk.doc>.

[21] Michael Parsons (West Coast Labs), Personal Communications, 5 August 1999.

[22] Virus Bulletin, <http://www.virusbtn.com/>.

[23] Fraser Howard (Virus Bulletin), Personal Communication, 2 August 1999.

[24] ICSA, <http://www.icsa.net/>.

[25] Larry Bridwell (ICSA), Personal Communication, 2 August 1999.

[26] Vesselin Bontchev, *Methodology of Computer Anti-Virus Research*, Doctoral Thesis, University of Hamburg, 1998.

[27] *Sophos Data Security Reference Guide 1999/2000*, Sophos Plc, 1999, <http://www.sophos.com/support/docs/>.

[28] *Survivor's Guide to Computer Viruses*, ed. Victoria Lammer, Virus Bulletin Ltd, 1993.

[29] *Department of Defense Trusted Computer System Evaluation Criteria*, NCSC, 1985.

[30] Péter Ször, *Happy Gets Lucky*, Virus Bulletin, April 1999, pp 6-7, <http://www.virusbtn.com/VirusInformation/ska.html>.

[31] NetBus, <http://www.netbus.org/>.

[32] BO2K, <http://www.bo2k.com/>.

[7] This paper is an updated version of [1] & [2].

[33] Systems Management Server (SMS), <http://www.microsoft.com/smsmgmt/default.asp>.

[34] *It has been reported that…*, <http://www.netbus.org/>.

[35] *Don't worry Windows Users, Everything will BO2K*, Cult of the Dead Cow, <http://www.cultdeadcow.com/news/pr19990719.html>.

[36] *What Customers Should Know About BackOrifice 2000*, Microsoft, <http://www.microsoft.com/security/bulletins/bo2k.asp>.

[37] Ian Whalley, *Talking Trojan*, Virus Bulletin, June 1998, pp 9-10.