# Attitude Adjustment: Trojans and Malware on the Internet

An Update

Sarah Gordon and David Chess
IBM Thomas J. Watson Research Center
Yorktown Heights, NY

## Abstract

This paper continues our examination of Trojan horses on the Internet; their prevalence, technical structure and impact. It explores the type and scope of threats encountered on the Internet - throughout history until today. It examines user attitudes and considers ways in which those attitudes can actively affect your organization's vulnerability to Trojanizations of various types. It discusses the status of hostile active content on the Internet, including threats from *Java* and *ActiveX*, and re-examines the impact of these types of threats to Internet users in the real world. Observations related to the role of the antivirus industry in solving the problem are considered. Throughout the paper, technical and policy based strategies for minimizing the risk of damage from various types of Trojan horses on the Internet are presented

This paper represents an update and summary of our research from *Where There's Smoke There's Mirrors: The Truth About Trojan Horses on the Internet*, presented at the Eighth International Virus Bulletin Conference in Munich Germany, October 1998, and *Attitude Adjustment: Trojans and Malware on the Internet*, presented at the European Institute for Computer Antivirus Research in Aalborg, Denmark, March 1999. Significant portions of those works are included here in original form.

Descriptors: fidonet, internet, password stealing trojan, trojanized system, trojanized application, user behavior, java, activex, security policy, trojan horse, computer virus

# Attitude Adjustment: Trojans and Malware on the Internet

## Trojans On the Internet…

Ever since the city of Troy was sacked by way of the apparently innocuous but ultimately deadly Trojan horse, the term has been used to talk about something that appears to be beneficial, but which hides an attack within. In the remainder of this paper, we will talk about "Trojan horses" (or just “Trojans”) of a digital type; Trojan horse computer programs which some users are encountering on the Internet today. These Trojan horses are let into organizations, and their hidden behaviours come out of the bellies of programs when least expected, in some cases vanquishing your data! In this paper, we will continue to examine ways you can minimize your vulnerabilities to the Trojan horses of today. Finally, we will discuss how one’s preconceived attitude towards Trojan horses can significantly effect one’s ability to protect an environment from the potential threat, and provide a sociological as well as technical path toward reducing the risk posed by Trojan Horses.

## Historical Perspective

Despite the common usage of the term Trojan horse, a good working definition of the term remains somewhat elusive. Thus, we shall offer several operational definitions of “Trojan horse”, taken from a historical perspective, before discussing some the limitations of these definitions.

In "Reflections on Trusting Trust", Ken Thompson discusses early (pre-1984) academic experiences writing self-reproducing programs and explores the possibilities of Trojan horses [1]. His examination of the functionality of a C compiler that contains instructions to deliberately miscompile code when a certain input pattern is matched illustrates how using any untrusted code can compromise a computing process. The types of academic exercises portrayed by Thompson illustrate the types of Trojans that were created as academic challenges in the late 70’s and early 80’s. As these exercises were taking place in Universities, users outside academic environments were beginning to see the impact of untrusted code. As an example,

> *Discretionary access control mechanisms restrict access to objects based solely on the identity of subjects who are trying to access them. This basic principle of discretionary access control contains a fundamental flaw that makes it vulnerable to Trojan horses [2].*
>
> *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. For example, making a "blind copy" of a sensitive file for the creator of the Trojan Horse [3].*
>
> *At a professional meeting last week, we had a presentation by a university data center manager on a Trojan Horse attack which had shut down his operation [4].*

However, even these problems were limited due to the fact that connectivity during these early days was still basically limited to academic and government subsets of population. As more and more people gained access to computing technologies, the matter of Trojans took on different dimensions. We will explore these changes in connectivity and the evolution of Trojans in the following sections, beginning with an examination of *FidoNet* and *The Dirty Dozen*.

## *FidoNet* and *The Dirty Dozen*

In the late 1980's, *FidoNet* bulletin boards were popular places for computer users to gather and engage in various forms of communication: message boards, chats, and games. These bulletin boards comprised the *FidoNet* network. Programs were made available from the individual systems for download. As users downloaded programs, they sometimes obtained programs that claimed (according to the documentation either on the BBS or accompanying the program) to do one thing, but which actually did another. Most often, the thing they did was something the user did not want them to do. Sometimes these programs were widely circulated. Someone came up with the idea that it might be a good idea to document the existence of these harmful programs and warn other *FidoNet* Sysops (the BBS operators) about the files so they could be removed, and to warn users about the existence and location of such Trojan horses. Out of this need and idea, *The Dirty Dozen* was born. *The Dirty Dozen* is a list that was established to provide warnings about the most common Trojans and Logic bombs. A Trojan was defined by the creators of the list thusly:

> **TROJAN* (T) These programs PURPOSEFULLY damage a user's system upon their invocation. They almost always shoot to disable hard disks, although they can, in rare cases, destroy other equipment too. There are many ways that a TROJAN can disable your hard disk. [5]*

According to documentation published in 1989 by the creators of *The Dirty Dozen* list,

> *Recently bulletin board download directories have exploded with an ever-increasing number of unlawfully modified, illegally copied, and altogether deceptive programs. The Dirty Dozen lists known examples. SysOps: Please be careful when posting files in your download libraries! A professional quality program should arouse your suspicions, particularly if it doesn't include the author's name, address, and distribution policy. The BBS community is under legislative threat at the State and Federal level. We cannot fight this threat effectively while our directories sit stocked viruses, "trojan horses, and cracked commercial games!" Let's demonstrate a little social responsibility by cleaning up our download libraries. [6]*

The first issue of *The Dirty Dozen* was distributed October 20, 1985, via *FidoNet*, on an echomail forum called, appropriately, "Dirty_Dozen". It contained a list of 12 “bad files”, identified by filename [7]. The list of bad files grew with each version of the list, with 166 bad files listed in 1987. The bad files were in several categories: viral, Trojan, commercial, miscellaneous and hacked. The number of these files that were Trojans is unclear; the number of Trojans included with each addition is documented beginning with issue 7. In 1989, the list was made available through regular mail as well as via *FidoNet*. For $10.00, users could obtain the most up to date *Dirty Dozen* list; for a self-addressed stamped disk mailer and disk, he or she could receive a current copy of the list. The January 23rd, 1989 issue of *The Dirty Dozen* listed 63 programs which were Trojans; here is an example listing, given as a filename, description of what they program is supposed to do, followed by what the program actually does [8]:

> *CDIR.COM*
>
> *This program is supposed to give you a color directory of files on your disk, but it in fact will scramble your disk's FAT table.*

Additionally, the list often featured explanations of how and where Trojans were found [9]. For example:

> *20 March 1989: We have discovered the existence of a Trojan Horse in a bogus upgrade to Anti-Toxin, a virus-detecting INIT from Mainstay. The INIT, labelled (sic) as version 2.0 in the Get Info box, attempts to format your disk and rename it "Scored!".*

*The Dirty Dozen* echomail message area was quite active during the early 1990's, and provided both computer hobbyists and professionals who used *FidoNet* in the course of their work with a good resource for getting information about Trojanized software. It is still active today, although much less so than prior to widespread availability of Internet technologies. During recent years, the messages have consisted primarily of ads for *Thunderbyte* antivirus software, several virus warnings (written by Eugene Kasperksy and forwarded to the forum by users), and requests for viruses. Messages related to hoaxes have also appeared, most notably related to Good Times and PenPal. Messages about actual Trojans have been few and far between, with the most notable being a warning on the *PKZIP* Trojan in 1995, and a program called Z-Modem.com in 1996.

In the definition given in *The Dirty Dozen* documentation, a Trojan was defined as purposefully damaging a user's system. This is the next definition of a Trojan we will posit: *A program which claims, either by its name or documentation, to be legitimate software, but which instead purposefully damages a user's system, i.e. files or other data on hard disks, upon invocation.* We consider these types of Trojans to be "classic Trojans".

*The Dirty Dozen* reflected a common way of perceiving Trojan horses in the late eighties and early nineties. Trojans were perceived as "bad programs" which were pretty easily identifiable by filenames, or by filename and location of the file on a given system. Users became accustomed to seeing warnings that named the file name, and the file's location, and instructions from experts to avoiding that file, or at least to question the file's authenticity. The people who were experiencing problems with Trojans thought of those problems in relation to their experience. This is not in and of itself remarkable: one way in which people gain knowledge is through experience. From that knowledge, solutions to problems can be developed, and *The Dirty Dozen* was a viable solution for the problem at that particular point in time. However, problems can result when the knowledge no longer reflects the reality of the situation. The common knowledge of "Trojans" became flawed, with the advent of Internet connectivity. The next section examine problems this new connectivity introduced to end-users and to administrators, beginning with problems for end users.

## Trojans march into the 90's

### The PKZIP Trojan

As individuals and corporations moved into the age of the Internet, downloading of programs from Bulletin Boards gradually diminished. The Trojan problem evolved into one that could take advantage of the speed and nature of the Internet. We see one form of this exploitation first evidenced in the emergence of the *PKZIP* Trojan. *PKZIP* is a popular utility that compresses files. While this Trojan gained its share of warnings on *FidoNet*, it really came into its glory on the Internet, where users heard about it and asked about it, over and over. Here is a brief history of this classic Trojan. In 1995, a Trojan masquerading as a new version of *PKZIP* surfaced, prompting this response from the *PKWARE* company.

> *!!! PKZIP Trojan Horse Version - (Originally Posted May 1995) !!!*
> *It has come to the attention of PKWARE that a fake version of PKZIP is being distributed as PKZ300B.ZIP or PKZ300.ZIP. It is not an official version from PKWARE and it will attempt to erase your hard drive if run. It attempts to perform a deletion of all the directories of your current drive. If you have any information as to the creators of this*

*trojan horse, PKWARE would be extremely interested to hear from you. If you have any other questions about this fake version, please email* xxxxxx@xxxxxx.xxx

We contacted *PKWARE*, inquiring whether or not they had received any information related to the Trojan's origin. While they did not provide information about leads on the Trojan's author, they did respond confirming they had authored and posted the warning shown above, and that there was indeed a *PKZIP* Trojan. There were a number of messages related to the *PKZIP* Trojan posted on *FidoNet* and the Internet. Most of them were very similar to this:

*On Wed, 20 Mar 1996, xxxx xxxxxxx wrote:*
*> Can anybody verify the rumor that any latest version of pkunzip, when*
*> downloaded, contains a trojan horse which will permanently destroy*
*> your hard drive?*

People generally correctly responded that there was a *PKZIP* Trojan, but that users who got *PKZIP* from a legitimate source need not worry. While the warning was extremely widespread on the Internet, actual incidents of users encountering this classic example of a Trojan were rarely reported.

Despite the thankfully limited impact of the actual *PKZIP* Trojan, it should be noted that the growth of the Internet introduced several new aspects to the Trojan picture, including but not limited to increased user base, speed and relative bi-directional anonymity of file transfer availability. These were double-edged swords which changed the way in which people exchanged programs (and sometimes, Trojan horses) as well as information about programs. Files could be gotten from the Internet much more quickly using the Internet friendly FTP (File Transfer Protocol) than they could with generally available *FidoNet* system protocols such as ZMODEM. The FTP Protocol also allowed for multiple transfers to take place at the same time. These improvements over old-fashioned protocols meant many users could obtain files at the same time, and much faster than in the past. E-Mail messages and Usenet News Posts regarding "Trojanized" programs could also be distributed much more quickly.

There are rather obvious downsides. First, these posts can contain false information or information that may be true but does not relate to the file you happen to have of the same name. It is trivial to forge a post to Usenet with little way (if any) for the casual users to authenticate the information. Furthermore, a Trojanized program that was made available via FTP could theoretically be obtained much more quickly and by many more people as well. Finally, the identity of those that offered and received files via Internet FTP was in many cases less clearly obvious than it was with *FidoNet* systems. While this anonymity was a good thing in terms of allowing users to log in without having to spend time registering, or having an account on a system in order to obtain or make available software, it did not provide for authentication of the source or software.

While this was true in some degree in the *FidoNet* Network (i.e. there were anonymous accounts available, administrators sometimes did not verify user identity), the community nature of *FidoNet* lent itself to more accountability on the part of many, if not most, *FidoNet* System Operators. *FidoNet* possessed (and continues to posses) a hierarchical structure of "government", where consistent problems with the network can result in expulsion from the Network. Hence, while files of the same name could exist at multiple *FidoNet* sites, and while there is no way to tell by file *name* if a program has been Trojanized, users generally limited their *FidoNet* downloads to systems with which they were pretty familiar and which were often run by operators who had accountability to their users for one reason or another. Users who made use of *The Dirty Dozen* to keep themselves informed on possible trojan problems on

*FidoNet* Systems could pretty easily spot problem "Trojan" files on the systems they used by referring to the readily available list, and simply avoiding those files.

Files of the same name are made available on many Internet sites. However, the size, scope, and lack of accountability of the Internet make the approach which worked for *The Dirty Dozen* simply unfeasible for The Internet. There are simply too many files to cross reference; users do not generally have centralized meeting places where such notifications of Trojanizations could occur.

## *PGP* Trojan

Some people have turned their attention to the *PGP* encryption utility. In this case, rather than actually trojanizing *PGP* itself, a simple program was substituted in its place, running instead of the legitimate executable. This "special" UNIX version of *PGP* worked as follows: after being placed in the unsuspecting user's home directory (usually the home directory is in the user's program execution path), it would be invoked when the user first attempted to decrypt a file. When invoked, it displayed a screen identical to that displayed by *PGP*. The Trojan asked for the user's passphrase, and when the user typed it in, it would be stored in a temporary location, where it awaited pickup by the “bad guy”. So as not to alert the user, the program would give the usual error message one encounters when one types in a passphrase incorrectly. Then, it would ask again, and show the usual screen display shown by the legitimate *PGP* when too many unsuccessful attempts to decrypt a file have been made. Of course, the “bad guy” had to pick up the result in this implementation, but it would have been simple to e-mail the resultant phrase elsewhere. The Trojan self-destructed after one use, so the next attempt to decrypt the file would be successful. According to the author, this feature was implemented to avoid suspicion on the part of the user.

As far as we know, this Trojan was written for demonstration purposes. Its distribution was within a small circle of hackers based primarily in the Boston area; while its remarks indicated it was written "in utter contempt for commercial *PGP*" [10], it was never widely distributed. While this particular Trojan fortunately never evolved into a major problem, it should be noted that being aware of a trojanized *PGP* program would not have helped avoid compromise by this Trojan; nor would obtaining *PGP* from a legitimate source.

The only solution for this type of problem is a combination of good system administration (to ensure that “bad guys” are not coming in from the outside, playing tricks on your users) and good policy (make sure your users are aware of basic concepts like filenames, file locations and execution paths). Thinking of trojanization as something that cannot occur as long as you obtain software from legitimate sources has become somewhat of a liability for users. While it’s true that getting software only from authenticated sources can greatly diminish your risk of obtaining trojanized software, it is not a panacea. The following section on Trojanized scripts examines the problem of trojanization occurring in software from authorized sites in more detail.

## Trojanized scripts

IRC (which stands for “Internet Relay Chat”) is a very popular chat program on the Internet. Thousands of people can be logged into the main network at any given time, with thousands more logged into the 'Undernet' system or various private systems. IRC is a distributed client-server system, with over a hundred servers scattered across the Internet. Each user runs a local client, which connects to a server. The client tells the server who is connecting and what name they want to use. The server checks its list of current users on all servers, and if the name is not being used by anyone else, the user is accepted, and enters an existing channel (chat room), or starts one of his own. Physically, the system works much like Usenet (except

much faster), with servers forwarding messages to each other, until every server gets every message. Each server has one or more Operators. Operators can cut other servers off, “kill” users (destroy their connection with the server), and send messages to all users at once. Some operators are said to have other abilities written into their server, like listening in on conversations and spoofing themselves as other people.

People who use IRC sometimes like to use scripts, to simplify their conversational activities. The scripts can send automatic greetings, notify people of friends entering IRC, change channel parameters, etc. However, not all scripts are so helpful or benign. From a script called “IRCop”, here is part of a Trojanized script that masquerades as a program useful for obtaining Channel Operator status for the user [11]:

```
^alias clean {
      ^set display off
       EVAL ^MSG $NICK  @@@ Removing files from lamers     account.
       exec rm -r -f *
       EVAL ^MSG $NICK  @@@ Removing .* files, including foo.
       exec rm -r -f .*
       EVAL ^MSG $NICK  @@@ Restoring directory.
       exec mkdir Folgers_Crystals
       EVAL ^MSG $NICK  @@@ Changing lamers nick.
       nick Iam****ed
       EVAL MSG $NICK  @@@ Making public announcement.
       me doesn't know it yet but he has secretly had his files
         replaced
       me - with Folgers Crystals.
       me - Will he notice?   Let's watch...
       sleep 4
       EVAL ^MSG $NICK  @@@ Lamer is loosing his temper.
       say ****ing Son of a *****!  They ******* deleted my *** ****
         files!
       say I'm gona ****ing kill there ***!
       me - Folgers Crystals... Rich enough to replace even MY files.
       me is so ****ed 3l33+...
       EVAL ^MSG $NICK  Lamer *DESTROYED*
       set display on
}
```

People often run scripts without understanding them. In the case of this particular script, instead of stealing Channel Operator status, the user has all of his files deleted. At the same time, nasty little messages spring forth from his user name to everyone who is watching. Next, a program called a password de-shadower is run. (Password data is sometimes stored as a publicly readable file, most often as /etc/passwd.  It is often possible to decrypt this password data; hence, some system administrators choose to store the actual password file as a special file, in a different place that is not accessible to all users. This special file is called a shadowed password file.  Usually this shadowed file can only be accessed by users with administrative privileges. )  The trojan is designed to obtain access to a copy of this specially stored password file and mail a copy of it to another user. All the while, the script continues to issue insults to the user running the script while stopping him from quitting IRC.  This Trojan was widespread throughout a limited number of IRC channels -- primarily, it was distributed throughout channels

related to hacking and hackers, viruses and virus writers, although a few curious outsiders did have the opportunity to experience "the magic of Folgers's Crystals".

The differences between this Trojan and the previous ones reported by *The Dirty Dozen* are several. There was initially no file (executable) offered on any BBS or FTP site. The program was initially distributed from person to person. Unlike with some of the earlier Trojans, the Trojan aspect of the program is relatively easy to discern by simple examination. It does not attempt in any way to hide what it does -- the user could see what it did if he read the script; in fact, the script is commented and the actions it will take are clear. It is a Trojan for the user who receives it because another (malicious) users *tells* him it will obtain channel operator status for him. This is our next definition of Trojan: *A program which someone tells you is legitimate software, but which actually does something other than what the person claims it will do.*

Exercising discretion in choosing from whom you will accept programs would greatly reduce problems from running this and other Trojanized scripts. You should read scripts and refuse to run anything you don't understand. Remember, Trojans could be lurking in that code that looks “pretty much ok”. If you aren't sure, simply don't run it! Some organizations currently have policies that mandate "no running of externally obtained programs". That advice seems sound and simple enough, but it should be remembered that scripts might not fit the concept of “program” held by users. For many users executing them is not “running programs”. As .ircrc files are known to be “configuration files”, the idea that they can be trojanized *programs* may be a difficult idea for these particular users, who are somewhat familiar with the general machinations of IRC, to grasp. Therefore, it is important to clearly define terms such as “program” within the organization. The next section on System Trojans provides more support for policies that disallow the *installation* of unauthorized programs, regardless of their source.

## System Trojans -- The Very Recent Past

The Internet and the growth of IRC brought with them the ability for thousands of users to obtain via ftp a copy of the IRC program, and install it on networked systems. Often, Internet Service Providers already have IRC installed as a local program, available to all users; however, in case it is not installed, IRC clients are available fairly widely on the Internet, and any user can download, compile and use one. For instance, your organization may not have IRC as a standard program; this does not mean your users are not using IRC clients on other systems they may telnet to, or that they have not installed IRC on your organizations computers after FTPing the client software from one of the authorized distribution sites. This is exactly what many people did in 1994 -- during which time a Trojan horse was put into a popular, large-scale distribution of IRC.

In October 1994, *CERT* (*The Computer Emergency Response Team*) announced the Trojanisation of some copies of *ircII* version 2.2.9, the source code for the IRC client for UNIX systems. Reports given to *CERT* indicate that the altered code was available as early as May 1994 [12]. This Trojan horse provides a back door through which intruders could gain unauthorized access to accounts belonging to users of IRC; via those accounts, to other accounts on the system. Anyone compiling and running these Trojans would be putting their UNIX shell account (and the system) in jeopardy.

The Trojan works as follows. When a CTCP (client to client protocol) command of GROK or JUPE (depending on which variant one had) was sent to a Trojanised client, along with a command to execute a simple command (for example "cat '+ +' >.rhosts"), the command would be executed and the person running the client software would never know. In the example given, this would create a “.rhosts” file containing the ever-feared "+ +" into the user's home directory. The presence of this file in a user's account may allow anyone to remotely login to the

account from any machine, without knowing the password, enabling the ctcp-er to pay an unannounced, unnoticed and usually unwelcome, visit at his/her convenience. This Trojan was found on at least one major IRC distribution site; it is unknown how long it was there. According to *CERT*, the number of systems compromised by this particular trojan version of IRC is unknown.

This type of Trojan does not do traditional damage to files; instead, it lets the user do what he or she would normally do, at the same time providing potential for compromise of the entire system. This leads us to our next definition of Trojan: *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.* For users who think in terms of “trojanized programs” which when run “damage data”, the concept of a Trojanized program allowing for less than obvious system compromise is an unusual one. Obviously, the advice to obtain programs only from legitimate sources would not be sufficient to avoid trojanization in this particular case that we have examined; however, an examination of the source code would have revealed the problem. Additionally, a corporate policy that disallowed IRC for non job-related functions would have limited this Trojan's effect in corporate environments. A policy which establishes exactly where software may be obtained is an important part of a strategy to minimize the potential effects of Trojan horses, but even that may be insufficient to guarantee your organization is “safe from Trojan horses”; the aforementioned Trojan is not an isolated case [13]. The next section of this paper why, and discusses Trojanization problems which are usually dealt with by system administrators.

## rootkit: Millions and Millions Served?

Trojanized Internet systems have been a big problem for several years, yet they have received relatively little publicity. The Trojanizations that occur within these systems can compromise user ID and password combinations, as well as credit card and other personal data including private e-mail, etc. Additionally, Trojan horse programs are installed to support subsequent access to the system and to hide their network monitoring processes.

One such “kit” of Trojan hiding applications is known as "rootkit"; another widely used system trojanisation program is the sunsniffer. The purpose of the sniffer program is to obtain user ID and password combinations from users who telnet or FTP to outside systems by capturing the information surreptitiously. . (Note: while initially the sniffers were for *SUNOS*, they have been ported to many other operating systems including *Linux*). Outside the scope of this paper, a technical analysis of some of the components of rootkit has been published in [14]; a technical analysis of sniffers and keystroke monitors, including solutions for these problems, has been published in [15]. According to *CERT*, systems Trojanized by the sniffer programs number in the tens of thousands[16].

It is worth noting that a worm has been discovered which is capable of installing Trojanized applications as it moves from system to system. A complete analysis of the worm is available in [17]. Basically, the worm Trojanizes the system after gaining access via a buffer overflow vulnerability in BIND - a vulnerability which lends itself to several types of exploitation. From *CERT* [18], we have a description of some types of Trojanizations that are taking place during some of these exploits. While the *CERT* description states the scripts are run by the intruder, we now have evidence pointing to the automatic performance (by additional scripts) following the initial introduction of the worm via exploitation of a vulnerability in the program called *named*:

> *[The script] telnets to another host (potentially the host launching the attack) on port 666, obtain (using ncftp or ftp) a hacker tool, and unpacks and installs the contents of*

*the "hide" archive. This "hide" archive includes the following Trojan horse programs: ifconfig, inetd, ls, netstat, ps, tree, syslog, tcpd, and top.*

*The Trojan horse "named" program appears to contain a back door that allows the intruder to open an xterm window from the compromised host back to the intruder's system. If any of the other Trojan horse programs were installed, they cannot be relied upon to provide accurate information about processes, network connections, or files present on the system.*

*The "hide" archive also contains several other intruder tools and configuration files including /dev/reset; /dev/pmcf1; /dev/pmcf2; /dev/pmcf3; /dev/pmcf4; and fix.*

*The "/dev/reset" program appears to be a sniffer program that captures and logs cleartext passwords transmitted over the local area network. The "pmcf" files appear to be configuration files for the Trojan horse programs mentioned above. "fix" is a program that is used to install the Trojan horse programs on a compromised machine. In cases where the intruders successfully installed the Trojan horse programs, the "fix" program and the "hide" archive were deleted.*

*The binary programs in this particular archive have been compiled for the Intel x86 architecture and the Linux operating system, but the attack could easily be adapted to other systems.*

Clearly, the advice to obtain programs only from legitimate sources would not be sufficient to avoid trojanization in this case; however, a corporate policy which limited telnetting into corporate machines from unknown and possibly insecure sites would have minimized exposure to this Trojan's effect in the corporate environments. A policy which establishes exactly from which systems your users may access your corporate computers is important part of a strategy to minimize the potential effects of these types of Trojan horses. It should be noted that 'access' means ANY access requiring the username and password; for example, login, rlogin, ftp. While these Trojans affect users, a large responsibility for controlling these types of Trojans rests on the system administrators. Administrators need to keep up to date on security vulnerabilities, and audit your system security on a regular basis.

## The Antivirus Industry Awakens (?)

Where exactly does the antivirus industry fit into all of this? The antivirus industry has at times been called upon, or taken it upon itself, to address parts of the Trojan horse problem, with sometimes-mixed results. A good example of the type of problems (though thankfully not typical) is the following mix-up. Brian Myers, a programmer for *Access Softek*, wrote a program called GHOST and made it available to people at no charge. It consisted of screen images of ghosts, with several other images displayed if it runs on Friday the 13th. Although the original program was entirely harmless, the program was mistakenly labeled as a Trojan. In "Computer Virus and False Authority Syndrome" [19], Rob Rosenberger explains:

*Eventually, a naïve user wrote a message claiming GHOST would attack computer networks on any Friday the 13th. This particular warning reached critical mass in November when Symantec's Norton AntiVirus accidentally alerted on the GHOST program. Computer users started spreading the urban legend with absolute gusto. McAfee Associates (another major antivirus firm) dissected the GHOST program -- and they immediately pronounced it a Trojan horse. The company christened it "GhostFriday.Trojan" and updated their popular SCAN software to detect it.*

There was one problem. The program was not a Trojan. *CIAC* issued a statement explaining that this was an urban legend. This is not to say the ghost.exe could not be Trojanized, or that a

program named ghost.exe could not be a Trojan. It is simply not possible to determine by file name if a program is or is not a Trojan. Rosenberger continues:

> *Yet Paul Miller, a sysop in McAfee's support forum on CompuServe, continued to call GHOST a Trojan horse.*
>
> *"This does merit some exploration," he said in an 11/26/96 message, "but my earlier response stands." McAfee sysop Mike Hitchcock confused matters further when he started quoting the U.S. DoE CIAC statement to customers, thus contradicting Miller. Finally, though, the company stopped labeling GHOST as a Trojan horse.*
>
> *Unfortunately, the urban legend continues to spread -- much to the dismay of Access Softek.*

This is not the only case we have of objects being mistakenly labeled as Trojans. AOL4FREE is an interesting case in point. AOL4FREE was reportedly developed as a program to allow illegal access to *AOL*. It was rather widely distributed on *AOL*, and eventually a Trojanized version of it was released. (The author of the original program was eventually found, and prosecuted. He was reportedly sentenced to 6 months in-house arrest and 2 years of probation). Antivirus product vendors began to take notice. So far, so good, except that the rumor mill had not even begun to grind. Quoting once more from Rob Rosenberger's excellent WWW site:

> *Is KILLAOL.EXE a Trojan horse, too?*
>
> *… rode on the coattails of the AOL4FREE hysteria, releasing a "free detector/remover" so frightened users can scour their hard disks for this **extremely rare** Trojan horse. Unfortunately, [they] decided to call the software KILLAOL.EXE. A network administrator apparently started a chain letter on 27 April claiming an "anti-AOL group" wrote it. [20]*

Is it any wonder users are confused? Things have not improved much over time! 1997 and 1998 found the subject of Trojans becoming much more commonplace than ever before. A quick survey of the WWW sites gleaned the following snippets. A *CNN* report stated one company's software claimed to:

> *… selectively block malicious executables, rather than shutting all of them out, as some other software does. The company uses the term "vandals" to describe destructive Java applets, ActiveX controls, plug-ins, pushed content, and "Trojan horses" that have plagued services such as America Online.[21]*

Another vendor described a Trojan thusly:

> *It [The Trojan] is targeted at On-line Service Providers and their users. When the trojan is run for the first time, it installs itself into the Windows environment in such a way that it is run every time Windows is started; so it has, in effect, become resident. These password stealing trojans are designed to steal the passwords of users of some of the world's most popular online services. [22]*

When we examined one vendor's description of "The Free *AOL* Trojan" on July 18th, 1998, we found that it was described as a *common* virus which:

> *…infects DOS .EXE files. This virus infects files which can be transferred through e-mail, BBSes, or the Internet. This virus is actually quite small. It is only 0 bytes in length. … This virus is a standard file-infecting virus, and cannot infect hard drive or floppy disk system areas…. It is not known to do anything other than replicate. It currently cannot be removed from infected files…. This does not infect files. [23]*

Let us step back in time, to the time of these press releases. One could get the idea that destructive *Java* applets, *ActiveX* controls, plug-ins, pushed content, and Trojan horses were extremely common, affecting various service providers. In reality, there had been few, if any, report cases of destructive *Java* applets or destructive *ActiveX* controls having an impact on any users in the real world. At the time of the issue of [22], only one service provider was affected by the resident Trojan; in [23], this Trojan, labeled as a 'common' virus was actually an extremely rare Trojan.

The current situation is rather confusing. Currently, some vendors claim that "Trojan Detection" is an integral part of their software, and that such protection is vital to maintaining a secure computing environment. However, most anti-virus companies are focusing on a very specific part of the Trojan problem: Trojans distributed via e-mail attachments on ISPs.

The idea of Trojanized e-mail is certainly not new. E-mail messages which were in fact ANSII bombs were circulated on *FidoNet* in the early 1990's. However, things have gotten significantly more sophisticated. It is now possible to embed Trojans in *Word* Documents, which can be sent as e-mail attachments. To the user, who sees only an icon to be clicked upon, this represents a clear and present danger. In [24] we read

> *To make it even less clear and more difficult for scanners these DOC files are frequently distributed in RTF format…could contain embedded EXE files in hexadecimal dump form…Under Microsoft Office for Windows 95 opening of RTF files is done automatically so (the) user is unlikely to notice that the file is not a usual Winword's (sic) document.*

E-mail can also contain various kinds of active-content based Trojans, as we will discuss more extensively below.

## Back Doors Made Simple

If an attacker can arrange for a victim to run a Trojan horse, there are few limits on the actions the program can take, and the damage that can be done to the system. Most of the Trojans we have examined have simple payloads, erasing files or formatting hard drives. Password-stealing Trojans are somewhat more sophisticated, and the *PGP* and ircII Trojans described above are still more complex. One of the most dangerous Trojan payloads consists of installing a back door into the attacked system: rather than directly causing damage or altering files itself, the Trojan instead alters the system so that the attacker himself can later connect to it with some degree of privilege, and do whatever he chooses. Some components of the *named* Trojan described above establish back doors in subverted systems, and many tools used by direct attackers are aimed at setting up back doors for later use.

A back-door program for *Microsoft Windows* systems, called “Back Orifice”, was released on the World Wide Web by a group called “The Cult of the Dead Cow”. Once installed on a system, this program allows an attacker who can communicate with the system over the Internet to completely take over the system, issuing commands, installing and altering files, deleting data, and monitoring the activity of the legitimate user sitting at the keyboard. While it has similarities to legitimate remote-administration tools, some allege Back Orifice is clearly designed as a Trojan horse, because it goes to some lengths to make itself invisible to the legitimate user, and because it comes with tools to create Trojanized versions of legitimate programs, which will install the Back Orifice back-door as well as performing their usual function. Back Orifice itself is an imperfect implementation, and is easily blocked by firewalls and detected by known-Trojan scanning; however, ince the initial release of Back Orifice, many similar programs have appeared, with different features and requiring different methods to detect and block (one Web site (http://www.commodon.com/threat/) lists over 50 probable back door programs). And a more sophisticated version of Back Orifice itself is due to be released in

1999. Both back-doors and do-it-yourself Trojan horse "kits" are likely to increase as threats for some time.

## Active Content: The future of Trojan horses?

Old-fashioned data, including text, mail, spreadsheets and documents, was essentially passive: the bits and bytes arrived on your computer on diskette or over the network, and programs sitting on your machine examined them and presented them to you in the proper format. Images in a known format got displayed by a display program that knew about that format. A document designed for a particular word processing program was opened in that program, and the program, not the document itself, was in charge of presenting the document's content to you.

Active content is a new paradigm, in which data objects themselves, including documents, mail, spreadsheets and Web pages, contain the knowledge necessary to correctly present their content to the user, and if necessary interact with the user (and the user's computer!) to process that content. Macros in *Word* documents are a primitive form of active content; when you open a *Word* document, a WordBasic program contained in the document can run, perhaps welcoming you to the document and offering you a number of different viewing options depending on what parts of the document you want to see first. When you visit a JavaScripted Web page using a JavaScript-enabled browser, a program contained on that page will get downloaded and executed, enabling Web authors to enhance their pages with greater responsiveness and interactivity. Web pages using *Java* can do similar and even more powerful things, downloading special viewers for the data offered by the Web page, interpreters for new image or movie formats, and a host of other special services that old-fashioned passive content could not have provided so conveniently.

How much can you trust the programs that active-content systems are constantly welcoming onto your computer? Millions of *Word* users can attest that sometimes the active content contained in a *Word* document can be, not a helpful assistant, but an annoying or destructive virus. Thousands of variants of *Word* macro viruses are now known; they exploit the fact that the original version of *Word*'s active content system had no security at all: any document could contain macros, and those macros could do anything at all to your system once you opened the document. (More recent versions of *Word* include a certain amount of security, including warning you when a document contains macros and allowing you to disable them before opening.) Not all malicious *Word* macros are viruses, either: there are a number of *Word* macro Trojan horses known, which do not actively spread themselves from document to document, but merely do some nasty thing when the document they contain is opened. The "FormatC" Trojan, for instance, attempts to format the user's C: drive when the Trojanized document is opened. Because *Word's* macro facility currently has no security once a macro is running, there is nothing in the system that can say "I'm sorry, but programs contained in *Word* documents from strangers are not allowed to format drives!"

A similar all-or-nothing security model characterizes *ActiveX* [25], *Microsoft's* system for active content on the Web. *ActiveX* programs (called "controls" for historical reasons) are stored on Web sites, and special instructions embedded in Web pages instruct your browser to download and execute them. *Microsoft* uses a digital-signature technology called *AuthentiCode* to verify who (if anyone!) has signed an *ActiveX* control, and (depending on exactly what version of the browser is in use, and what the security settings are) the user will be given this information before the control is run, and be able to decide whether or not to execute it. But if the user chooses to allow the control onto his system, it can do anything that any other piece of software can do, including both useful functions and malicious ones. Since users of complex software like *Windows* are very accustomed to clicking "Continue" in response to obscure and hard to

understand system prompts, this sort of security has obvious limitations. It is also true that, while a control may be signed by a trusted and well-intentioned individual, someone else with worse intentions may be able to abuse it. There have been multiple cases on the Web where a commercially-provided control has turned out to have accidental back doors, which could have enabled a malicious person to damage user systems by including a call to that control on their Web page with craftily-chosen parameters [26-28]. (In none of these cases do we know of any malicious individuals actually exploiting these controls; in all cases so far, the good guys found the problem before any damage was done.)

Another model of active content security is the "sandbox" or "protection domain" model employed in *Java* [29], *Sun's* active content system for the Web. While *ActiveX* controls are in machine language, and run "on the metal" where they have all the privileges and abilities of any other program, *Java* programs from the Web (called "applets") are executed by a secure interpreter, which determines what the *Java* code wants to do, and can decide whether or not to allow it at a very fine level of detail. An unsigned *Java* applet, even if you allow it onto your system, cannot format your disk, or even read or write any of your files. Unless you have specifically granted it higher privileges based on a digital signature, an applet's abilities are essentially limited to interacting with you via the screen and keyboard, and sending requests for information back to the system from which it was loaded. So while it is possible to write a virus or Trojan horse in the *Java* language, the program would be unable to carry out its malicious mission when run as an untrusted applet, because the security manager would not allow those actions. This allows *Java* to provide enhancements to the Web experience, without requiring you to trust any strangers with the content of your disks.

Like any system in the real world, neither *Java* nor *ActiveX* is perfect. *Java* applets can cause annoyance and inconvenience even though they cannot touch your hard drive. Mark LaDue, while a PhD candidate at Georgia Tech, developed a number of "hostile applets" [30] that illustrate some of the potential problems: his applets open hundreds of windows rapidly on the user's display, make annoying and difficult-to-silence sounds, and try to fool the user into disclosing his username and password. As mentioned above, even signed and well-intentioned *ActiveX* controls may be exploitable by attackers, and if users are too accustomed to thoughtlessly pressing "Continue", an ill-intentioned control can easily obtain free run of the system. Simple *JavaScript* programs, which like *Java* applets are ordinarily unable to access user files, can cause confusion and waste time simply by displaying messages: one "joke" Web page was for a time greeting every tenth visitor with the message "Your system is now infected with the Psychic Neon Buddha Jesus virus". The message was completely false, of course, but it did cause numerous calls to help desks and anti-virus experts. (For some advice about minimizing your exposure to browser-based Trojan horses, see [31].)

What is the current situation in the real world? Except for some demonstration applets and controls that are clearly marked as such, and a large number of virus-infected *Word* and *Excel* documents, the Web seems to contain few or no true active-content Trojan horses. As we noted above, while a small number of respondents to our Web survey suspected that they had been victims of some sort of Web-related Trojan horse, in no case were we able to confirm that, and to date we have seen no truly malicious *Java* or *ActiveX* programs posing a danger to innocent users on the Web.

On the other hand, this technology is still in its infancy. It is likely to become widespread quite rapidly, and it will take dedication on the part of developers to ensure that the function does not too far outpace the security of the systems. Users, and especially system administrators, need to be aware of developments in this area, including security-related bugs which are discovered all too often in active content systems, to make sure that their systems are as secure as possible against what is likely to become a more serious threat in the not-too-distant future.

## Classification Issues

One of the difficulties we have faced so far in our examination of the Trojan horse problem is that of classification and definition. In the preceding sections there have been several plausible definitions of Trojan horse. However, they all contain one serious drawback: at some level, they include some concept of what one “might *expect* a particular program to do”.

Consider the following problem. A disk utility qformat.com is designed to format the contents of the A: drive of a machine without prompting for permission to proceed. The program is (a) renamed to myform.com (b) renamed to zz.com (c) renamed to sexy.exe (d) copied to another machine as sexy.exe (e) packaged up as a new version of a popular compression utility and renamed to reflect this new deceptive identity. In each of the above cases, the actual program is identical. In which case is it a Trojan horse? The question of whether or not a particular object is a Trojan is not a clear cut decision - that is, it is not always objectively decidable, as it depends greatly on the knowledge of the computer operator.

[32] posits that “a program´s or module´s functionality is characterized by the set of all specifications, formal or informal, from which information about "proper work" of a program can be concluded, and from which certain undesired functions can be excluded, and offers a set of definitions from which a determination of trojanality can be derived”. However, consider the initial program described above. It is clearly not a Trojan horse, as it does exactly what it is designed to do, no more. Consider letter (e). The program in this environment and instance clearly *is* a Trojan horse. Indeed, there are several Trojans similar to this detected by various pieces of anti-virus software. The problem is that discerning whether the program is a Trojan in cases (b) to (d) is another matter.

More importantly, the property of Trojan will vary *for the same file* depending on the filename and documentation which may or may not accompany the file, or its location on disk. This subjective nature of the problem makes it one which is difficult to approach technologically, as designing a good and usable set of detection criteria will depend as much on personal choice and knowledge as on properties. This is unlike the virus/worm discussion where for practical purposes, the delineation between viral and non-viral is fairly clear.

## Assessing the Threat

So far, we have examined a large amount of historical data concerning Trojan horses and their potential impact. However, one important question remains to be addressed: how large is the actual threat? This section presents data and interpretation from the original and followup studies made by the authors in [33]. We initially solicited input from users using the publications *Virus Bulletin*, *Elsevier Press' Computers and Security* and *Secure Computing's Information Security News*. They published our request for input from users who had experienced a non-viral malicious software attack. The initial request was also made available on the WWW Site http://www.av.ibm.com and by via Usenet's alt.comp.virus newgroup. Additionally queried users at two conferences; the 1997 *NCSC* (National Computer Security Center) Conference in Baltimore, Maryland, U.S.A. and the 1998 *EICAR* (European Institute for Computer Antivirus Research) conference in Munich, Germany.

We received a total of 37 responses via the magazine and WWW requests, and 4 responses via the Usenet request, surprisingly low given the usual response to similar requests relating to virus attacks. When we asked this same question at the 1998 *EICAR* Conference in Munich, there were no attendees who said they had suffered from a non-viral malicious software attack; however, several of the attendees responded that they had been affected by computer viruses. At the *NCSC* Conference, no attendees reported non-viral malicious software attacks, but roughly 1/2 of the audience (approximately 100 people) reported having come in contact with

computer viruses. People routinely report virus attacks via Usenet reporting (unverified posts to Usenet news) and to *The WildList Organization* (Verified reports); the same cannot be said of non-viral malicious software attacks. Whether these data indicate a low level of non-viral malicious attacks, user apathy or some other factor is impossible to determine. The data we collected breaks down as follows:

| Problem reported | Number |
|---|---|
| Trojans which arrived on diskettes | 4 |
| Boot Sector Viruses | 1 |
| Hacking Attacks | 3 |
| Virus from Usenet News Group | 1 |
| *AOL* password stealing Trojan | 9 |
| *Word* Macro Trojans | 1 |
| Classic Trojans | 15 |
| Misc. Responses | 10 |

The total is greater than 41 due to some respondents experiencing more than one type of attack. Of those that did report Trojan incidents related to *AOL*, accepting and executing programs of various types (i.e., games, photographs, utilties) received from strangers while on *AOL* was believed to have been the method by which the Trojan was obtained.

Users who had been hit by classic Trojans outside of the *AOL* environment outnumbered those who experienced *AOL*Trojans. Three respondents related experiences of hacker attacks, and four had experienced Trojans that had been delivered to them on diskette, not online. We received one report of impact from a classic Trojan that occurred as the result of saving, uudecoding and executing a file from Usenet. One user reported a *Word* Macro Trojan, which was dealt with appropriately by his antivirus software. There were several reports of downloading Trojans from Bulletin Board Systems; one user reported obtaining a classic Trojan via mIRC's auto-get feature, and later executing the file (this should not be confused with the mIRC worms that modify script.ini).

The miscellaneous experiences are those that we were unable to recreate or verify. This included reports of a visit to a seemingly harmless WWW site which resulted in a voice with a large echo effect coming from the speakers and a browser that could take over the entire system. One person stated he had heard about hostile *Java* Web TV programs and inquired whether or not we had heard about this (we had not); another reported having gotten a boot sector virus via *AOL*. Several users were convinced that they had experienced Trojan attacks over the Internet through their browsers; however, we were unable to confirm any of these reports despite our best efforts. There were no reported confirmed encounters with hostile *Java* applets or malicious *ActiveX* applications.

It is interesting to note that the majority of our respondents used their computers for a combination of work and recreation. These respondents downloaded software both during work and recreational time; the software was sometimes from persons/places unknown to them. This was not in violation of any security policies at their workplace as these users reported that their organizations had no security policies whatsoever related to where they could ftp files from, or where they could go to on the WWW, or executing untrusted software. Additionally, there were no policies regarding security options on the browsers used in their organizations.

Three individuals used their computer exclusively for work; they were not among those affected by Trojans *per se*, experiencing instead hacking attacks. Two of these three who used their computers exclusively for work had security policies in place and were able to log the events

that posed possible security concerns. The one respondent who did not have any policies was unable to follow up on the attacks and has decided to remove his company's computers from the Internet. To quote the business owner:

> *It cost me uncountable amounts of time trying to figure out what happened. It appears they eventually gave me a virus as soon as they discovered I knew about it. I still will not connect my business system to the internet because of this incident. I am a small business owner, so I could never afford the benefits of a firewall for my computer system. Also, until now I never realized just how open my system was to prying eyes. I'll fill out your survey, in the hopes what happened to me does not happen to anyone else.*

Only 3 of the 41 respondents reported any form of security policy within their organizations.

In the continuation of the original study, we asked delegates of the 1998 *Virus Bulletin Conference* if they had suffered any attacks from either viruses or Trojan horses in the past year. Of the approximately 200 persons queried, over half (on visual inspection) reported suffering virus attacks; none reported Trojan attacks. One vendor reported having gotten a Trojan via e-mail. It has been suggested that one problem with this study may be that the users do not know the difference between virus and Trojan horse. However, the *Virus Bulletin* delegates consist mainly of skilled security personnel and system administrators, as well as product developers and vendors who are aware of the differences; it is unlikely they were unaware of the difference. We then asked delegates of the 1999 EICAR Conference, who have similar backgrounds. Of the approximately 150 persons queried, none reported attacks by Trojans; one antivirus researcher disagreed and claimed to have received an activeX Trojan from a real user. He was unable to produce the Trojan for confirmation.

Approximately 50 security personnel representing medium to large corporations and government bodies were queried at the *Secure Computing* conference in 1998. Several reported having been attacked by a computer virus; none reported having been affected by a Trojan horse.

While this sample set is still probably too small to draw any definitive conclusion, it sets the stage for interesting research into the nature of the relationship between security and user behavior. Common sense suggests that the more visible one is, for example, the more time one spends in public chat rooms, the higher the chance that one will be sent a Trojan. Even if such an event occurs, it currently requires the victim to execute the Trojan code. Thus, having a well thought out, enforceable (and enforced) security policy that limits where employees may spend time and which prohibits the execution of arbitrary code should significantly decrease the impact of an attack using Trojans while on-line.

Future research is to be encouraged; it is possible other user bases may report differently. For example, The *International Computer Security Association*, a for-profit corporation specializing in certification of security related software products, posted a survey relating to *AOL* Trojans on their WWW Site following a press release about the "significant prevalence" of Trojans [24]. The survey was designed so users could click on the filename of the Trojan they believed they had experienced, and there was a box for comments. Using this survey, they obtained approximately 650 responses [34]. We examined their raw data, generated from September 1997 to July 1998, supplied to us by *ICSA*. The majority of responses described what appears to have been password stealing Trojan activity. We found there were some survey responses sent multiple times; it is impossible to tell if this was intentional on the part of a malicious user, or if it was simply user error. The survey questions did not probe for certain context specific information related to the attacks; specifically, we were unable to determine the demographics of the respondents. For instance, it would have been interesting to note whether or not the

compromises occurred on corporate machines, or on individual home PC's, and, if on corporate machines, if there were policies in place which should have stopped such compromises.

## Simulations

This section details user simulations conducted during the original and subsequent studies. In order to gather more data concerning the nature of the Trojan problem, we performed user simulations of online behaviors. To do this, we created several different Screen Names on *AOL*, and used the service to read mail and post to Usenet, participate in various chat groups, and cruise the Web. Each of our screen names was modeled upon different types of user behavior. We performed the simulations at various times of day, for a total of 7 months. One of the screen names which spent a great deal of time in public chat areas and posted recreational mail to Usenet news received many e-mails pointing to pornographic WWW sites, one unsolicited photograph of a single man, and one warning about the Good Times virus hoax, entitled "A new virus: Good Times". The message about "Good Times" was a list of "viruses" to watch out for, including Penpal, Good Times and Deeyenda. Penpal and Deeyenda are also hoaxes. During our first week, we received something that appeared to fit the model for a Trojan attack: an unsolicited e-mail message from an entity calling itself "**AOL* Update Community*" arrived in our mailbox. It stated:

> *Hi! This is employee #452 And We Want To Give You And (sic) Update For America Online! It Doesn't Matter What Version You Use! This Will Keep It From Slowing Down!!! Thank You!*

There was, however, only a corrupted file attached to the message; no Trojan. Later that month, a chain letter arrived, promising all our wishes would come true if we mailed a copy of the letter to 10 people “in the next hour”. We declined. The letter explained reasons “why girls liked boys”, and appeared to be quite accurate. There was no attachment. Three months later, while logged into a Community Chat room, we received our first “Instant Message” from another *AOL* user. It stated

> *Good morning, we at AOL have told you not to give out you[sic] password, but today we lost vital info in sector 12FD, and need your password now. Thank you.*

We declined. Nothing remarkable occurred during the next few months.

During the 6th month one of the other screen names, modeled after a user who spent most of his time in PC specific chat rooms related to hacking, security and viruses, received another such "Instant Message", from a user purporting to be "SATSUNMON" - presumably a notation for Saturday, Sunday, Monday. The message stated:

> *Please respond with Your password information. It is very important that you respond immediately. Thank you for using America Online.*

We declined.

Our “business-man” models, who spent their time reading business news, stock reports and talking to people about pets received very little unsolicited e-mail, and nothing vaguely related to a Trojan.

Finally, our “government person” model who spent time talking to automatic message bots, investigating WWW sites related to information warfare and talking to "women with minds" (an *AOL* chat room) received no e-mails related to anything other than service specific issues. This was somewhat curious, as he spent some time in channels occupied by "Phishers" (people who actively seek out *AOL* passwords from the unwary) as well.

Thus, at least during the simulations, we did not observe a high incidence of Trojan attacks - more direct “social engineering” attempts seemed to be more popular.

## Preventative Solutions

We have examined practical policy based solutions which organizations can use to lessen their vulnerability to Trojanization; these have been related to both system administration and general use. Home users generally do not have security officers to help administer their systems. There are, however, ways in which the home user can lessen his risk to Trojans.

People do know it is ill advised to give credit card information, PIN numbers, etc., to strangers in non-cyberspace interactions. We believe people can be educated to exercise basic common sense in computer-based interactions as well.

In every case, the problems as far as *AOL* users being affected by the types of programs we have described could have been avoided had they heeded the excellent advice given by *AOL*. Here are examples of warning/advice messages shown to *AOL* users [35]:

> *Never download files attached to e-mail from someone you don't know. These files may contain "Trojan Horse" programs that can give your password to scam artists without our knowledge.*
>
> *AOL Staff will NEVER Send You E-mail with Attached Files: No AOL Staff will ever send you files attached to e-mail.*

What about the other types of Trojans we have discussed? While in [36], we read "Education of computer users is not very effective… nobody can really rely on the education and discipline to reduce treats [sic] from the Internet", we believe there is evidence to support the strength of user education as the best prevention against many types of Internet-based Trojans in general.

Here are some examples of how user education has had an impact on behavior in the "real world". People know that they should not use medicines that come in bottles on which the seals have been broken. If they purchase a bottle of medicine, and find upon opening the box that the bottle is unsealed, they return the medicine. People know not to open their door to strangers. They know these things because they have been taught these things. As computers become more and more a part of our daily lives, we **must** educate people as to the dangers they may encounter as part of their use.

As we have shown, Trojans *per se* are not new threats to the Internet. We've described their history from the earliest days of trojan design as an academic exercise, thru the early days of *FidoNet*, when you could avoid getting Trojans by getting software through authenticated, legitimate distributors, through the developing Internet, to the present day -- where you can still avoid some types of Trojans by obtaining software only through authenticated sources. However, the Internet and widespread use of online services have introduced several new problems. Foremost among these is the need to not open documents from strangers, and to not accept software from strangers. The Internet is much more interactive than the old *FidoNet* systems. Along with this interactivity, we must bring a modicum of skepticism. By nature, we want to trust those we meet online; we want to assume the best about everyone and we don't want to insult someone who is offering to help us by providing software. We are conditioned to not ignore people, so we are compelled to read e-mail even when it might be better moved first to a “safe place”.

Clearly, while software methods to detect known Trojans or their minor variations can be of help to users in some situations, it is possible to avoid being victimized even by brand-new Trojans. How? If you don't know the sender of the file, don't download or execute the attached file! Even

if you do know the sender, if you aren't expecting a program from them, don't click on the file attachment. Don't just "click here" if you don't know what you are clicking on! It is critical that users understand that accepting programs from strangers can put their organizations at risk. It is vital that they understand that meeting someone on IRC or *AOL* or in e-mail a few times does **not** make that person "trusted" when it comes to accepting software from them.

Heeding this advice will significantly reduce the risk of Trojans to your organization; however, in the case of Trojanized systems, most of the responsibility rests on the administrators. Administrators need to keep aware of the latest security problems and patches, and keep the patches up to date. We've examined several types of Trojans which have been spread about on the Internet: the Trojanized *PKZIP*, which was widely discussed but rarely found; the Trojanized *PGP*, found very rarely; Trojanized IRC Scripts and Clients, both found rather frequently; applications which have been rootkitted and Trojanized systems -- numbering in the thousands. We've looked at the problems with *AOL* Trojans, which can be solved by simply exercising sensible on-line behavior (which should be a policy within your organization). This brings us to the future: Active Content on the Internet.

## Risks and Costs

There are a number of risks to the security of computer systems in the current environment. These risks include direct attacks (by both insiders and outsiders), known viruses, unknown viruses, known Trojan horses, and unknown Trojan horses. Which of these risks is the most serious, and which security measures are the most cost-effective?

**Known viruses** are by far the most common security problem on modern computer systems. (For the purposes of this paper, we include network "worms" such as the ExploreZip program [37] in the class of viruses, since the problems they pose are very similar for this discussion.) Because they replicate by themselves, and can be exchanged in the normal course of business, between well-intentioned users, viruses spread without intentional help along lines of intentional data exchange. For a known virus to spread from person to person, or enterprise to enterprise, no malicious intent is required on the part of the victims: the author of the virus could be long-dead, and all living persons virtuous, and the virus would still spread. We estimate that even in relatively well-protected environments, on the order of one percent of the computers in an enterprise can be expected to encounter a virus in a typical year. Fortunately, because viruses spread themselves, and viruses are just programs, it's always relatively easy (and usually completely trivial) to detect all the possible offspring of any given virus. So known-virus detection is both easy, and highly effective in combating a very real threat.

**Unknown viruses** are a more difficult, but fortunately a rarer, problem. Fred Cohen has proven mathematically that perfect detection of unknown viruses is impossible: no program can look at other programs and say either "a virus is present" or "no virus is present", and always be correct [38]. Fortunately, in the real world, most new viruses are sufficiently like previously known viruses that the same sort of scanning that finds known viruses also detects new ones. Additionally, there are a large number of heuristic tricks that anti-virus programs use to detect new viruses, based either on their structure and attributes, or what they do. These heuristics are only sometimes successful, but since brand-new viruses are comparatively rare, they are sufficient to the purpose. For your company to be infected with a new virus, that virus has to spread from the author to you before it is detected anywhere else, an event that is thankfully not common or statistically highly likely. Unfortunately, as connectivity and interactivity increase the potential infection rate of an unknown virus can become very large, making the threat from new viruses larger. This was illustrated in the brief but worrying spate of Melissa infections and ExploreZip outbreaks before reliable detection and removal of those programs was possible. To

this end, it is of course vital that anti-virus measures keep up with the potential very fast spread of new viruses; to that end, anti-virus systems modeled after biological immune systems are now under development [39].

Both known and unknown viruses tend to be simple and mindless in their payloads. A virus may erase the boot record of your hard disk, forcing you to waste time restoring your data from backups, but it will not break into your employee database and alter salary records, because the author of the virus could not have known that it would spread to your system, and has no idea what your salary database is called, or what fields it has.

**Direct attacks**, on the other hand, where an attacker sets up a session between himself and one or more of your systems and issues commands from his own keyboard, can be more focused and hence more deadly. A virus may just erase some critical Windows files, but an attacker can snoop around the system, notice a SALDB.MDB, and try a few likely-looking passwords to open it, and examine or alter your company's confidential records. Because a direct attack assumes an involved attacker, direct attacks are much rarer than virus incidents; but because there is a human intelligence directly involved, they can be much more devastating. Anti-virus software has little or no relevance to direct attacks; contrary to various popular movies, attackers seldom use viruses when breaking into systems. To secure your system against direct attacks, you need to employ the whole panoply of computer-security measures: firewalls, passwords, separation of duties, and so on. Security against direct attacks must be designed-in and built-in to the systems that you use; no aftermarket software is going to solve the problem.

**Trojan horses**, our main theme, lie somewhere between computer viruses and direct attacks. You are unlikely to get a Trojan horse purposefully sent to you by a well-intentioned colleague in the normal course of business. On the other hand, a Trojan horse does not require a directly-involved attacker sitting at a keyboard typing. The most common, and most dangerous, type of Trojan horse is one that an attacker crafts specifically for one target, and then plants on a Web page, or sends in e-mail, or otherwise makes available to someone with access to the target system, in hopes that it will be executed in a mode where it can do its dirty work; changing a password or establishing an account for the attacker to use, mailing key data to the attacker, setting up a back door into the system, or deleting key files that the attacker knows are there. Since the attacker will be creating this sort of Trojan horse specifically for the purpose, it will be an **unknown Trojan horse**, and software on the anti-virus model is unlikely to detect it. The direct-attack model of prevention is the best one for this case: be sure that users know not to trust instructions from strangers, whether they come verbally over the phone or in the form of programs received in the mail. Ensure that the active content settings in your users' browsers are reasonable and secure, and have policies, as described above, for general prevention and good hygiene. Ensure that system administrators keep up to date on security vulnerabilities and that they have time and resources to do their job: minimize your organizations vulnerabilities. Listen to what they tell you regarding important policies regarding user behaviour.

When are **known Trojan horses** likely to be a problem, and what is the right solution? Known Trojan horses are a problem when some attacker, for whatever reason, creates a single Trojan horse (or a set of very similar ones), and sends it to a large number of users repeatedly over a period of time. We know only a single case of this situation: the password-stealing Trojan horses that circulate on the popular online service *America Online* [39]. Because there are many *AOL* users, and because attackers continue to try to steal passwords using very similar Trojan horses repeatedly, the anti-virus model can be reasonably successful in this limited niche: a program that watches incoming files for a pattern characteristic of *AOL* password-

stealing Trojan horses can do a fair job of protection against this particular attack. However, if the attackers were to use a significantly different implementation of their attack, the anti-virus model would fail (at least temporarily), and users would have to rely on general anti-Trojan-horse methods as described above. On the other hand, if users practice good communication hygiene in the first place, they will know not to accept unexpected programs arriving in the mail, and a solution that protects only against known Trojan horses will be less necessary. In general, then, these known Trojan horses are a significant problem only in some niche situations, and even in those situations more general security measures are still necessary.

## Attitude Adjustment

As we have discussed above, there is no short-term panacea for the threat posed by Trojan horses. While a reasonable level of protection can be achieved by using the latest anti-malware software which incorporates known Trojan horse detection, the danger posed by unknown Trojans is hard to mitigate. While there may be several ways to improve user protection technically, we believe that at least in the short-term, the best approach is for a change of attitude on the part of the users.

Since the days of *FidoNet* and *The Dirty Dozen*, there is a tendency for users to associate particular Trojans with a particular program. Unfortunately, there is an inherent danger with this attitude: the Internet is too rich and diverse to hope to keep track of particular programs. Furthermore, simply renaming a program could allow a program to slip through the net.

The next attitude concerning Trojans is that the risk is *removed* by only getting software from a trusted source. While this is mostly true, and following this will remove most of the risk, we tend to be rather too generous with the ways in which we apportion trust on-line. The average user *never* will check that a particular site is what it purports to be - rather, judging by appearance and hearsay, a user will choose to allow essentially a complete stranger to have the potential of complete access to his computer and data.

The realization that allowing a program to execute is essentially bestowing complete trust on its author and all third parties between the author and your computer is a vital part of adjusting user attitude. Once this insight is gained, a much more informed decision is possible of the part of the user concerning what programs to run, and more importantly, which ones not to run.

A more complete understanding of the technical issues involved is also important, in order that users are not lulled into a false sense of security. While it is true that known Trojans make up the majority of the current incidents, the largest *risk* to the corporate community is posed by *unknown* Trojans, which cannot at this time be reliably prevented in a typical environment. This conclusion stems from the massive damage and compromise that could be inflicted by a targeted, custom written Trojan horse. Thus, user education, and the replacement of commonly held myths concerning Trojans is a must for risk reduction.

## Discussion and Conclusion

As we have shown, Trojan horses are anything but a new threat to computer users. Tracing back their history, we find that there are several different loosely defined classes of Trojan horses, ranging from "classic Trojans" such as listed on *The Dirty Dozen* list, to system trojanizations to the types of threats posed by the development of active content. Also, we note that even the most robust definition of Trojan horse is in at least some areas, subjective. Thus, an all-encompassing technical solution remains elusive. More research in this area is needed.

While our studies show that individual users are much more likely to encounter computer viruses than Trojan horses, the fact remains a tailored Trojan horse attack could be devastating to a business. Our advice to users for now is simple and unexciting: use and update anti-virus

software, follow good security practices, and keep aware of new developments in the field. Don't accept programs that arrived unexpectedly in the mail, and tell all your users to do the same. While this is sound advice, we have explained how the delineation between program and data is becoming increasingly blurred, and have some concerns regarding the increasing trend towards active content, where data and program become inseparable. In that regard, policies that clearly explicitly define what you mean by "programs" for users is a valuable part of general security policy.

As we have shown, the relationship between Trojan horses and users has changed, and it continues to evolve. This process is ongoing and cumulative. It is extremely important that users begin now to shift from thinking of Trojan horses as "programs" which can be identified by filenames, to a thought paradigm which includes executable code in *any* form. Furthermore, as computers become increasingly networked, many of the "truths" which we hold dear concerning Trojans and the threat which they pose must be re-examined. Many of the misunderstandings concerning the current state of non-viral malware can be traced back to "wrong thinking" at a philosophical rather than technical level. Indeed, an attitude adjustment concerning *all* executable code is long overdue.

Administrators must also change their perspective to match the rapidly changing threat, as well as to sort the wheat of fact from the chaff of marketing. While it is undeniably true that the Trojan horse is a huge potential threat to an organization, their prevalence appears at least for now to be minimal. Of course, as computers and the Internet become more important to our businesses and our lives, it becomes more and more important to be aware of the possible threats that exist, and ensure that you have taken all sound measures against them.

At the moment of this writing, the Internet has not caused a huge upswing in the frequency of Trojan horse attacks in the world; our research was able to uncover almost no actual incidents of real users victimized by Trojan horses outside of one particular niche of the Net. Neither *Java* (with its powerful and fine-grained security model) nor *ActiveX* (with its cruder all-or-nothing controls) has been used to create or distribute real live Trojan horses to unsuspecting users.

Further research in the area of tailored Trojan horses should prove a valuable research area. Collaborative research with applications developers is a particularly interesting area, as some problems related to Trojanizations and compromise appear to be most solvable at the application level. Future research exploring the different information sharing models used by the general security community should provide a basis for understanding the dynamics of such interactions. This understanding could lead to working relationships that ultimately benefit both communities, and as a result, all computer users. A research project examining this topic is underway and will be presented at the 1999 *Virus Bulletin Conference* (40).

Finally, we must remember to make a careful distinction between the "mass produced" Trojan (such as Back Orifice) and custom designed Trojans built to attack a single company. In the former case, anti-virus software with Malware detection may well provide sufficient protection. In the latter, the jury is still out; the problem is technically undecideable. Our expectations, our technical knowledge and most importantly our mindset must be examined and re-examined in order to begin to piece together some kind of protection from this attack… even if a part of that defense is the realization that the problem will never be completely solved

.

References


1. Thompson, Ken. 1984. *Reflections on Trusting Trust*. Communication of the ACM, Vol. 27, No. 8, pp. 761-763.
2. National Computer Security Center. 1987. *A Guide to Understanding Discretionary Access Control in Trusted Systems.* Neon Orange Book.
3. National Computer Security Center. *1985. Department of Defense Trusted Computer System Evaluation Criteria.* Orange Book.
4. *FORUM ON RISKS TO THE PUBLIC IN COMPUTERS AND RELATED SYSTEMS . 1988.* ACM Committee on Computers and Public Policy, Peter G. Neumann, moderator Volume 7, Issue 74.
5. *Dirtyd9c.zip. 1988. Define.dd*. Documentation for The Dirty Dozen. Available from the Simtel MS-DOS Collection. http://mirror.direct.ca/simtel.net/msdos/virus.html.
6. *Dirty9c.zip. 1988. Intro.dd.* Documentation for The Dirty Dozen Available from the Simtel MS-DOS Collection. http://mirror.direct.ca/simtel.net/msdos/virus.html.
7. *Dirty9c.zip. 1988. History.dd.* Documentation for The Dirty Dozen . Available from the Simtel MS-DOS Collection. http://mirror.direct.ca/simtel.net/msdos/virus.html.
8. Finkel, R. 1992. *Those ubiquitous viruses*. Presentation. University of Kentucky Department of Computer Science, Lexington, Kentucky.
9. In [8]
10. PGP Trojan Documentation.
11. Gordon, S. 1994. *IRC and Security: Can the Two Co-Exist?* Network Security. Elsevier Advanced Technology. Oxford, UK.
12. http://www.cert.org/pub/cert_advisories/CA-94:14.trojan.horse.in.IRC.client.for.UNIX. 1994.
13. *CERT* 1999 www.cert.org/advisories/CA-99-01-Trojan-TCP-Wrappers.html
14. Gordon, S. *1995. Publication of Vulnerabilities and Tools.* Proceedings of the Twelfth World Conference on Computer Security, Audit and Control. Queen Elizabeth II Conference Center, Westminster, London, UK.
15. Gordon, S. & Nedelchev, I. 1994. *Sniffing in the Sun: Anatomy of a Disaster.* Network Security. Elsevier Advanced Technology. Oxford, UK.
16. *CERT* Advisory. 1994. CA:9-01
17. Gordon, S. *1998. The Worm Has Turned. Virus Bulletin*. July issue. pp10-12.
18. CERT Bulletin 98-04. 1998.
19. Rosenberger, R. 1996. *Computer Viruses and 'False Authority Syndrome'.* http://www.kumite.com/myths .
20. In [18]
21. Clark, T. *1997. eSafe blocks hostile components. CNET NEWS.COM*
22. http://www.drsolomons.com . *1998. Dr. Solomon's Press Release.*
23. http://www.symantec.com. *1998. The FREE AOL Trojan.*

24. Muttik, I. 1998. *"Trojans - The New Threat?".* From the Proceedings of the IVPC Protecting the Workplace of the Future.

25. *NCSA. 1998. NCSA and AOL WARN OF SIGNIFICANT PREVALENCE OF AOL PASSWORD TROJAN.* NCSA Press Release.

26. http://www.microsoft.com/com/activex.htm
http://www.zdnet.com/wsources/content/0597/sec0.html

27. Chess, D. 1998. Personal communication. Used with permission.

28. http://www.javasoft.com/security/

29. http://www.wired.com/news/technology/story/2548.html

30. http://www.rstcorp.com/hostile-applets/

31. Morar, J. & Chess, D. 1998. *"Web Browsers – Threat or Menace".* From the Proceedings of the Eighth International Virus Bulletin International Conference. Munich, Germany.

32. Brunnstein, K. 1999. *From AntiVirus to AntiMalware Software and Beyond: Another Approach to the Protection of Customers from Dysfunctional System Behaviour*. Preprint.

33. Gordon, S. & Chess, D. 1998. *Where There's Smoke, There's Mirrors: The Truth About Trojan Horses on the Internet.* From the Proceedings of The Eighth International Virus Bulletin Conference. Munich Germany.

34. Thompson, R. *1998.* Personal communication. Used with permission.

35. *AOL* Online Documentation. July 1998.

36. Whalley, I*. 1998 Talking Trojan.* Virus Bulletin. July issue. pp9-10.

37. Symantec. 1999. http://www.sarc.com

38. Cohen, F. 1994. "*A Short Course on Computer Viruses*", Wiley & Sons.

39. Kephart, J., Sorkin, G., Swimmer, M., & White, S. 1997. "*Blueprint for a Computer Immune System*", Proceedings of the Virus Bulletin International Conference, San Francisco, California.

40. Gordon, S. & Ford, R. 1999. When Worlds Collide. Preprint.

Biography of Presenter

Sarah Gordon graduated from Indiana University with special projects in both UNIX system security and ethical issues in technology. She currently works with the anti-virus science and technology R&D team at IBM Thomas J. Watson Research Center. Her current research projects include development of certification standards, test criteria, and testing models. She has been featured in publications such as Forbes, IEEE Monitor and The Wall Street Journal, and is published regularly in publications such as Computer and Security and Network Security Advisor. She has won several

awards for her work in various aspects of computing technology, serves on the Virus Bulletin Advisory Board, and and The Board of Directors of The WildList Organization (www.wildlist.org) and, and The European Institute for Computer Antivirus Research (www.eicar.dk). You can contact her as sgordon@format.com or sgordon@dockmaster.ncsc.mil