Global Information Assurance Certification Paper

Advanced communication techniques of remote access trojan horses on windows operating systems

Candid Wüest

SANS GSEC Practical v1.4 option 1

January 16th 2004

Abstract

Trojan horses on Windows operating systems have been around for a long time. With exemplars like SubSeven or Netbus they were never taken serious and had been smiled at as toys for script kiddies. This paper will show the evolution of these malwares and explain why they can become a real threat to system administrators in the near future. New possibilities for communication methods will be outlined and the weaknesses and strengths of these techniques will be compared to the traditional techniques. With this, some possible attack scenarios will be illustrated and approaches for prevention will be discussed. Indicating what already can be done to protect from such attacks and pointing out where new methods need to be developed.

Contents

Chapter 1

Introduction

1.1 Definitions

The definition of a trojan horse or short form trojan varies depending on the source. A widely accepted definition is the one below from Rita Summers:

> A Trojan Horse is an apparently useful program containing hidden functions that can exploit the privileges of the user running the program, with a resulting security threat. A Trojan horse does things that the program user did not intend [1].

To make the difference between a trojan and a virus or worm clear, some characteristics have to be pointed out. A trojan horse does not replicate or distribute itself on its own. It does need user actions to start, usually this includes running the host program by intention.

Over the years many denotations have been created for different kind of trojans or related variations, like backdoor, rootkit, remote access trojan (RAT), keylogger, dropper to name a few. Most of them do miss a vital part of the above definition of a trojan horse. The useful feature of the host program is not present, if there is any host program used for the camouflage at all. Nonetheless we can see them as sub categories of trojan horses, as the basic idea of fulfilling a job hidden from the user is present in all of them.

For the rest of this paper we will concentrate on remote access trojans on Windows operating systems. From now on every allude of trojan will refer to remote access trojans on Microsoft Windows operating systems.

1.1.1 Backdoor

A backdoor is a program that opens a secret access to a system. Often used to bypass system security for easy access after a successful break in. Backdoor programs are very common in the UNIX world. After an attacker has broken into a system he will plant a backdoor or replace a daemon with a trojanized version. This for example can be a SSH login daemon which is altered to let any user in and grant him root privileges if the specified password is a predefined one, for example “31337”, regardless of later password changes.

1.1.2 Rootkit

A rootkit is a collection of files and scripts usually designed for a UNIX operating system or derivative. They include scripts to delete log files and modified versions of standard commands. These system scripts like the `ls` command are modified to hide any evidence of the presence of this rootkit. Making the system administrator believe that there is nothing wrong with his system. Rootkits are not widely used on Windows operating systems. They are more common on UNIX operating systems [2].

1.1.3 Remote access trojan

Remote access trojans (RATs) are typically client-server programms. They are doing a similar job like official remote control and management tools. Symantec's PCAnywhere can be named as an example for a remote control application [4]. The big difference is that a RAT installs itself hidden and runs invisible for the user. It gives an attacker full control over the infected machine as if he was sitting right in front of it. RATs are often used to upload and implant other malware.

1.1.4 Keylogger

A keylogger is often built into a rootkit or RAT. As the name implies it logs all key strokes made on a system, searching for passwords and secret informations. They sometimes act as a RAT letting the attacker remotely choose when to start recording and where to transfer the logged messages to.

1.1.5 Dropper

A dropper is a malicious program which will download or extract another trojan once executed. It is often a small application which is implanted by some other attack into the system. Used to put the foot in the door and later download the larger update from the Internet onto the system and runing it.

1.2 Motivation

Remote access trojans on Windows operating systems are not a new threat. With representatives like Back Orifice, which was released by a group called Cult of the Dead cow (cDc) in 1998, they gained some attention. Still the Internet took these tools as toys for script kiddies. At the beginning the features included mostly silly functions like opening and closing the CD-ROM. The stealth techniques used where very simple. Because the security awareness of users in those days was not as high as today, trojans where widely used to compromise systems for fun. Often playing a prank to friends. Most of these trojans where easy to find and to remove from an infected system when having the right informations. Because trojans where not detected by most systems, trojan authors did not need to develop new techniques to hide. Over a long time no real evolution took place. Trojans where used in a limited but still growing field to control systems. With the rise of the security awareness, trojan horse authors were challenged to evolve new techniques. In my opinion we are now at the beginning of this adaptation

to newer techniques. Practices from virus and worm coders incorporate with new developed methods of trojan horse authors. Bringing a head start from the protection tools. If we do not analyze this new techniques and develop counter measurements we soon will be confronted with many large trojan horse attacks. Therefore it makes absolutely sense to study trojan horses and learn to understand them.

1.3 Outline

Chapter 2 will introduce the common communication methods used by trojan horses today.

In Chapter 3 we will discuss the security software used to protect against trojan horses showing their strengths and weaknesses.

An overview of new attack techniques is given in Chapter 4. Explaining each method and illustrating possible ways to defend against them.

Chapter 5 will show how a possible scenario in the future could look like.

The conclusions gained in this paper are explained in Chapter 6.

Chapter 2

Standard communication techniques

In this Chapter we will see which standard communication techniques remote access trojans use today, by discussing the weaknesses and strengths of these methods.

2.1 Basic communication

A remote access trojan mainly uses client-server techniques to communicate and therefore consists of two files, the client and the server part. The server application gets installed on the victims machine and the client application is controlled by the attacker. The client can also be a telnet client or a browser. How the server is installed on the target system is a different story and will not be covered in this paper. The reader may think of ways like receiving it as an email attachment, distributing it over instant messaging systems or through file share networks.

To allow the communication the trojan horse server will open a listening port on the infected machine, enabling the attacker to establish a connection to it. To avoid conflicts with other installed applications and because of limited privileges, a non privileged higher port greater 1024 is normally used. In some cases it can also be the server part initiating the connection to the client. At the time when trojan horses began to appear, each trojan had its default communication port. Knowing the port number gave the possibility to identify an installed trojan by mapping the listening port to the corresponding trojan name. Nowadays every trojan has the opportunity to configure the listening port to any port the attacker wish, therefore making it impossible to identify the trojan's name by the port number. However there are still many old trojans running on systems in the Internet on their default ports. There are free lists showing the mapping of port number to trojan name available in the Internet, an accurate one is the one at Simovits website [3].

As the connection oriented TCP protocol is a lot easier to handle as the connectionless UDP protocol, trojan coders often use TCP communication between server and client application. Avoiding the hassle of dealing with reordering and lost packets required with UDP. Even thought that open UDP ports are more difficult to detect from remote and would thus provide more concealment.

When the connection is established an authentication password might be submitted. If implemented at all, this authentication method normally uses a hard-coded password

in the server that was configured before the installation of the server part. So far very few trojans use challenge response systems or other zero knowledge authentication methods. This is of course not a high priority task for a trojan horse author, as this password is only set to prevent others from overtaking this victim server.
The first goal for trojan horse writer is generally to remain hidden in the system. Once spotted and revealed the work of the trojan is finished. But exactly the hiding is not well done when using common communication methods as described above. A detection is not difficult with protection tools used today as we will see in Chapter 3.

2.2 Stealth mode ports

As pointed out in Section 2.1 a good starting point for detecting a trojan horse infection is to check for new listening ports on the local machine. Therefore it is also clear that trojan authors try to eliminate this possibility. One way to achieve this is to set the network card in a local state of promiscuous mode or to intercept the TCP/IP stack by hooking into it. It is not needed to listen to all the traffic on the network, only traffic directed to the infected machine is of interest. Having this setup it is possible to define a secret port sequence as a kind of password. For example an attacker can define that it is required to send a special crafted packet to TCP port 21, 23 and 80 in a time frame of 10 seconds and then the hidden communication port 4242 will be opened in listening state. With the help of this sniffer approach there is no open port that could be detected by a `netstat` command or by a remote port scan performed by the administrator. This would make this method really stealth, were it not for the promiscuous mode which will rise a huge red alert flag for every administrator. This is maybe also the reason why this method is not widely used and does not get much attention.

Chapter 3

Defense tools

This Chapter will illustrate the defense tools used nowadays on an end-user system to protect from trojan horse attacks, explaining where the strengths and weaknesses lie.

3.1 Anti-Virus tools

Even at home nearly all PCs have now an anti-virus software installed and running. Not all of them have it turned on the whole time or updated the anti-virus signatures regularly. Still as these products evolve, most of them offer a fair level of protection right out of the box with little configuration.

But having an installed and up-to-date anti-virus tool running, does not necessarily protect from trojan horse attacks. First of all, as the name implies anti-virus products where made to protect from viruses and not from trojans. This sounds kind of silly, but viruses and trojans are different as we remember from the introduction. This means that most solutions today will treat trojans as viruses and try to detect them with the same methods. As we all know one disadvantage of a signature based anti-virus product is, that it can only detect malware for which it has a signature for. Therefore it has to be captured sometime before and analyzed by a specialist. There is the rub. Trojan horses unlike viruses do not replicate themselves. Normally they are placed more or less systematicly on a finite number of targets and do not spread fast around the globe. The distribution may be very limited. One of this few exemplars has to be caught in the wild before a antidote can be made by the analysts. This can take quite a while until an instance of this trojan horse gets sent to a specialist. A virus which poses a threat will distribute itself fast over the global network and the chances of an anti-virus laboratory catching one instance of it raises with each infection. This fact makes it unlikely that an anti-virus product will detect a relatively rare spread trojan horse on a system even if the trojan is two years old.

There is even a second thing which makes the life of a trojan horse author easier. Different then virus writer they do not have to care much about the size of their trojan. These malicious pieces of software can easily grow to some hundreds of kilobytes without generating much problems at distribution. Therefore it is possible to reorder the source code, add some large junk sections and recompile the trojan. Resulting in a different binary that presumably will not be detected by an anti-virus tool as the signature for the original binary does not match. Sure, virus writers have used this technique

which is called polymorphism or metamorphis for years, but as we will see later in the discussion in Section 4.8 a trojan horse author has some advantages against a virus writer.

For completeness I should also mention that there exist a lot of tools in the Internet which make a piece of software that is detected by an anti-virus product undetected again. Simplest way is to use a hex-editor and change some bytes of the part which is detected in the signature. Another method is to use an executable file compression tool like UPX [7]. Latter will not render the binary undetected in memory as it will be extracted into memory in the old form. But a normal file scanner can be fooled with this tool.

This little excursion into basic polymorphism should show that an anti-virus solution will most likely not protect against trojan horse attacks, if they are performed by a clever attacker.

3.2 Personal Firewalls

Not yet on every end-user PC but gaining popularity we can see personal or desktop firewalls to appear. They range from simple inbound filters like the Windows XP built in connection firewall up to application aware stateful desktop firewalls which filter in- and outbound traffic. Some of them have IDS or anti-trojan tools included. To simplify matters I will handle a personal firewall as an in- and outbound application aware firewall and ignore the special features as they will be discussed separately in the next Section.

These techniques will provide a good protection against common trojan horses. As we have seen in Section 2.1 trojans often use TCP connections for communications. So as soon as the trojan horse server binds itself to a listening port on the victims machine the personal firewall will trigger an alert and block this attempt, as this unknown application is not configured to communicate with the Internet. As every unknown application is blocked by default, we will also block new trojans which can be unknown and maybe undetected by the anti-virus software. This brings the advantage of monitoring every communication attempt regardless of how well the source is hidden on the system. It might be hidden in an alternative data stream [5] not scanned by the anti-virus software but still the traffic will be blocked. Unfortunately there are some newer techniques targeting exactly this protection, helping the trojan horse to bypass the desktop firewall. This is what this paper is about and some of these methods will be covered in Chapter 4.

3.3 Anti-Trojan tools

As there are anti-virus tools, there exist also anti-trojan tools. One of the oldest on the market and sophisticated ones is Trojan Defence Suite (TDS) from Diamond CS [8]. Beside the common features like file scanning and memory space scanning which every anti-virus product today should provide as well, these special tools will further monitor the auto-start methods of the protected system. This is a very vital part for a trojan horse, as it needs to be somehow started when the system is started. The common

way used by trojans is to set a key in the registry for example at
`HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run`
Of course there are many ways to have an executable restarted with windows each time it reboots. Check the freeware tool “Autostart Viewer” from Diamond CS [9] which checks over 50 different auto-start locations. By monitoring these access points an anti-trojan software is sure to find interesting applications that need to be analyzed closer. If there is a common trojan installed, then it needs to use one of these methods to restart. Therefore the anti-trojan tool can use preciser heuristics as the file is suspicious and there are not that many files to perform a deep scan on. A scan over all files on the disk drive can then be done in a second step. Beside this, anti-trojan tools often perform file integrity checks on important system files. So that tampering will be detected, making it hard for a trojan to for example install its own network driver.

Anti-Trojan tools are a very good protection against trojan horses and can prevent from a lot of problems. Unfortunately these kind of tools or not yet widely used and have a shadowy existence. Today most people think that the anti-virus software they are using will protect them from all malware. This assumption might become true somewhere in the future but with the scanner technology of today it is wrong. To get a good overview what tools are on the market, have a look at the anti-trojan application test on Wilders website [10].

Chapter 4

Advanced techniques

This chapter will talk about communication techniques and methods used by a few trojans today and probably widely used in the near future. It will also point out defense methods against these attacks. This is by no means a complete list, but it does cover the most common techniques found during the research for this paper.

4.1 DLL injection

Dynamic link library (DLL) injection is the technique of having a malicious code in a DLL which will get loaded by another program. As most personal firewalls relay on trusted applications it is obvious that the primary target is to load a malicious DLL into a trusted application. Once in the same memory space as the trusted application this DLL-trojan can start to send and receive data from the network. As it will be treated by the personal firewall as the trusted host application no traffic will be blocked. Popular target for this attack is the Internet Explorer as it is present on all Windows operating systems and most user grant it full rights to access the Internet. It is a perfect target for an attacker to inject his code into it.

This method is not new and was widely discussed in diverse news groups. There exist already prove of concept applications like Firehole [11] for demonstration purpose. Source code snippets in various programming languages can be found on the Internet. Even ready made tools to include in projects exist as many websites demonstrate [13]. That is why personal firewall vendors have started to not only fingerprint the trusted application itself, but also include all the loaded DLLs for comparison tests. If one component changes, the rule will not be allowed. Once this detection method is used in all desktop firewalls this communication method will no longer be useful for an attacker, but until then it will work fine. Unfortunately not all that have already implemented a detection method for this attack, have done it properly as a Bugtraq article shows [12].

4.2 Process injection

Similar to the method of injecting a malicious DLL into another process it is also possible to allocate process space in a remote process and start a malicious thread in there. Often DLL injection is done this way by inserting a code segment which will then load

the malicious DLL. As the trojan will run as a child thread in the trusted application it will not show up as a new process in the task list. Even thought the thread itself is visible in the process list it will be hard to spot, as the names may be chosen at random. This method has started to gain more and more popularity among the trojan horse authors. Especially since there are ready to use source codes available on the Internet like on Madshi's home page [13]. Alike DLL injection this method targets trusted applications to misuse them later for traffic sending. Most methods used so far to protect against this attack will monitor remote thread creation calls and block them. By the time of writing this paper the author is not aware of a reliable method which works in a convenient way and flawless. Especially if the execution context from which the malware is launched has advanced privileges. This is often the case for home users. Therefore it is conceivable that a trojan will attack this vulnerable point of the security software.

4.3 Process killing

A method already used frequently by many trojan horses in the wild is the so called process killing. The trojan will attempt to shutdown the process of the security software. Thus rendering the system defenseless. Sending the "`Window Close Message`" will normally terminate the targeted process. A security application can however be configured to ignore this message. As always, there are many more ways to achieve a process termination. Just think of suspending all child threads leaving the targeted parent process frozen or modify the code in memory so that it will crash or exit. Many security softwares are not immune against all these attacks.

One side note at this point, killing a running application will hopefully make most users suspicious. If the familiar icon in the task bar is suddenly missing some doubts may raise. To prevent this, a trojan horse can place the original icon in a dummy process, so that the user is not missing anything.

Protection against these attacks is not trivial but can be achieved. Some personal firewalls like ZoneAlarm [6] do this by disabling the TCP/IP stack by default while the process is not running. Therefore making it required that the desktop firewall process is running when we want to communicate to the network. Other vendors have implemented guard processes which monitor each other and also block write access from other applications to itself. Still there is much room to improve security software's protection against these attacks.

4.4 Process modification

Being more reserved then simply killing the process, modification of a security application can also make the life of a malicious trojan easier. If the attacking trojan is able to write into the memory space of the personal firewall or anti-virus software then it might be possible to overwrite loaded rules in memory. Thus making it possible to disable the software or render it useless. Just think of adding an "allow all" rule at the beginning of a firewall or excluding all files from anti-virus scanning. To protect from such attacks a

security tool must assure that it is not possible to write into its memory space. Depending on the user context the trojan is executed in, it will not be possible to deny this write access completely. Monitoring and filtering those write calls can help here, but only if implemented from a lower system process. Otherwise the trojan will disable that monitor software first before attacking the anti-virus process.

4.5 Configuration changes

Let me explain this with a little thought experiment. Every security tool must have its configuration stored somewhere on the installed system. We will ignore enterprise versions which will download configuration files each time from the network. Think of home users. When they boot up their system the security tools will load the customized configuration. But who tells us that this is really the configuration we made ourselves and not a configuration made by a trojan horse? Ok right, the configuration is not likely to be in a plain text file lying around on the hard drive. One might say that they sure do protect the configuration files by checksums and maybe even by encrypting it. That is a clever idea, but remember the security software is able to load the configuration at startup. So it must be able to decrypt it. If the user changes anything at the configuration a new configuration file will be created and saved. So the application must also be able to encrypt it. Thus the encryption key must be present on the system as well. Maybe a registry key somewhere? Each version of a security software has its own secret place to hide the key and the encryption algorithm. But it has to be somewhere on the system as we have seen. What does hinder the trojan from encrypting its own configuration? Everything is present on the system. The only question is how to do it, though it is clear that it is possible. Some security tools make the work even easier as they deliver administration tools to generate such encrypted configuration files which are system independent.

Of course these changes will not be applied immediately, but the trojan horse could initiate a reboot to make sure of that.

The only possible way to protect against this kind of attack is to make sure that the trojan can not modify these configuration files. Thus these files have to be held under a single access lock by another process which is unkillable because of the reasons seen in Section 4.3. Running the security application with higher privileges can also bring more security for the configuration files. Just make sure that the application itself has no vulnerabilities which in turn could be misused to gain a root shell. Bear in mind that most home users work with an administrator account, hence having the files writable for administrators only might not be enough protection.

Some security products generate random keys during installation so that it is not possible to copy configuration files from one system to another. This is in contrast to the idea of having an easy manageable security environment in a company. A system administrator in a large company may want to be able to copy an image of the configuration file to all his client machines, without having to generate it separately for each machine. If this random key is stored in an easy accessible location then of course it is useless anyway. As the trojan will simply ready in this key and use the algorithm to encrypt its data as seen in the discussion above.

4.6 Protected stack bypassing

When sending packets to the network card usually the Windows own TCP/IP stack is used to generate the correct packets. This is just because it makes it easier to create the packets. A trojan can of course install its own network driver to generate its own communication channel to the network card. Most personal firewalls have hooked themselves in the Windows TCP/IP stack to filter all passing communication. When a trojan is using a different stack the desktop firewall will not be able to see the traffic and therefore also not block the communication. For testing purposes multiple prove of concept tools are available like Outbound [14].

To protect against this kind of attack it is best to drop all packets which are not generated in the standard Windows stack or filter all of them. Some personal firewalls have now started to include this feature in new releases.

4.7 Tunneling

All methods discussed above had one big disadvantage for the attacker. They allowed the trojan to bypass local security measures but not security devices on the way back to the attacker, like a network firewall. Therefore new approaches where searched, which could be combined with one of the other methods making it more concealed.

4.7.1 Simple tunneling

There are plenty of websites to this topic on the Internet [15]. A tunnel encapsulates a protocol inside another to transport other packets in its payload, for example HTTP. Tunneling is a general concept which can be used to carry a protocol across any network. It is not a new or secret method. In Virtual Private Networks (VPN) for example it is used for years to join two isolated networks with a secured link.

Any protocol can be misused for tunneling. The only requirement is, that the protocol should be permitted by the firewalls which will be passed on its way, as this is the general idea behind tunneling. Protocols like SMTP, DNS, ICMP and HTTP generally satisfy this requirement. This method will work against all packet filter in between. If special crafted packets are used they can even pretend to belong to an already established connection and might be ignored and unlogged by the intermediate firewall.

4.7.2 Advanced tunneling

Going one step further an attacker can even make the communication more sophisticated by disguising the payload. So the communication will really look like a normal HTTP, DNS, ICMP or what ever communication. An example therefor would be the DNS tunneling, where the trojan sends a request for `secretdata.passwords.attackersdomain.com` so the attacker gets the secret data, which of course can be encrypted. The response packet `192.168.42.23` would have new commands encoded into the IP address numbers. In our example the number 42 could

stand for start attack and the number 23 could indicate the target. This will be very hard to spot even for an IDS. If the implementation is clever made it will be impossible to distinguish this communication from normal traffic. Using a protocol tunnel communication will allow an attacker to bypass a firewall even if it uses an application proxy.

Another advantage of this feature for the attacker is, that the trojan makes a reverse connection. The communication will be initiated from the trojan "server" to outbound into the Internet back to the attacker. This method, sometimes also called insideout trojans, will eliminate the problem of network address translation (NAT) environments. Nowadays home users often have DSL modems with a NAT device in between. Therefore it makes little sense for a trojan to open a listening port on the target system and wait for incoming connections. The attacker will never be able to establish a connection from the Internet into this LAN segment. A simple way to accomplish a reverse connection is to hard-code a domain name from a dynamic DNS in the server application. (There are also some trojans with hard-coded IP addresses but this is obviously not really clever for an attacker).

A second method especially interesting if HTTP tunneling is uses, is to use a free webspace hoster like Tripod [17] to host a control site. The trojan server will, once having detected an active Internet connection, send a HTTP `GET` request to the attackers free website. There a special programmed script, for example a PHP site, will accept the data sent by the trojan server. This data may contain the current logged in user name, the local IP address, the hostname and any other information that the attacker thinks of being useful to him. The attacker can then send a predefined default command back in the HTTP response to the trojan horse server, disguised as a web page. Or depending on the data received decide to send something else back. Let's assume it is programmed to send back the command to start keylogging on the infected machine. The command is encrypted in the HTML file or in an image file that is send back to the trojan horse server. Which in turn, sends back a HTTP `POST` request with a huge encrypted keylog result file of all logged passwords. There goes the privacy right under the eyes of firewall and IDS. As this traffic looks exactly like a normal HTTP session, if not to say it *is* a normal HTTP session, no intermediate filtering device will block it.

One may point out that this kind of communication will not be suitable for real time screen capturing. This might be true, but why would someone want to have live video streaming from a victims machine anyway? This would give away an attack as it will produce a fair amount of traffic. If the attacker is not a script kiddie he will not need a screenshot of the machine in order to take it over. Some commands will be enough to install any tool and start a distributed denial of service attack from the infected PC to somewhere else.

A thing to remark on this scenario is, that even if we are able to detect the trojan and trace the communication back, we will end up at the attackers website of the free webhoster. There is never a direct connection from the victims machine to the attacker. This makes it quite hard to track down the attackers real origin. We would need the access log files from the webhoster.

4.8 Polymorphic code

A method nearly not used today in trojans is polymorphism or methamorphism. The technique of changing the appearance but leaving the functions unchanged. This is somehow surprising as virus writers use polymorphism since a long time and they even had to implement it using assembler code.

One huge advantage for trojan horse writers is, that they can generate polymorphic trojan generators which can generate thousands of different looking trojans. The big difference to polymorphic viruses is that the polymorphic engine does not have to be built into the trojan server. It can remain in the trojan generator and therefore the size limitation is not an issue and no protection from reverse engineering has to be done. If once an instance is caught in the wild and analyzed not all variants will be detected as they have different binary patterns but the same functionality. Metamorphic viruses provide no good ways to detect them. The same applies to metamorphic trojans. Heuristics and code simulation would probably be the best way to approach this threat. But as illustrated it would be quite difficult, as the trojan itself once generated would not be metamorphic anymore as it does not replicate itself. Of course this could also be implemented but is not needed. As the metamorphic engine will never be caught in the wild it will always be a guess how siblings of a trojan generation can look like. Making it nearly impossible to detect them on signature base. Therefore a signature for each instance has to be created, which is not feasible. Heuristics are needed to catch the trojan by its features and functions.

Chapter 5

Future scenario

In my judgment a scary but possible scenario in the near future will look like this. Let's assume hypothetically there will be a generation of remote access trojans which uses some of the discussed techniques. Most likely is a combination of these methods. My prediction is that they will use process injection techniques to bypass the local firewall. If no connection can be established process killing and modification methods are used to terminate the firewall process as fallback methods. The target process is the Internet browser. The final communication is made using HTTP tunneling to a script server somewhere in the Internet. All information is encrypted and disguised as normal HTML pages and image files. Thus rendering IDSs and network firewalls ineffective. There is no chance to distinguish this traffic from ordinary web traffic. Only the URL of the remote script can indicate the attack. Therefore multiple domain names with encoding are used for the same web page to confuse statistic analyzers and make it harder to filter.

To remain undetected on the infected system the trojan uses basic polymorphism techniques. This will generate a completely different looking binary for each installation. At a later stage modification of system tools will be used to stay undetected. Like UNIX rootkits, remote access trojans will hook themselves into API calls to filter any evidence of their presence. Making it hard for system administrators to find traces from them on infected systems. To avoid analyzing tools, anti debug techniques from virus writers are adapted. Rarely checked methods are used for startup. In some cases often used binaries are replaced completely by the trojan or infected like a virus. Every time this special application gets started the trojan will load itself into memory.

To stay competitive with security tools, the trojan can download updates from the master website. This makes it possible for the trojan author to develop an anti patch for his trojan. If the remote access trojan is updated before the security software gets the chance to apply its new patch, the antidote will be useless.

When used as a member in a distributed denial of service attack the source address of the flooded packets are spoofed to make it harder to find the infected machine in the network. No reasonable trace back is possible. Depending on the purpose of use, some worm features are added for wider distribution. Fast infection stays in contradiction to the guideline of attracting as least attention as possible. This behavior would be reasonable under certain conditions, like the mentioned distributed denial of service attack scenario.

Chapter 6

Conclusions

Today trojan horses are not taken as a big threat. Security experts often say that remote access trojans on Windows operating systems are baublery of script kiddies. More sophisticated attackers will implant kernel rootkits. It might be true that older trojans like SubSeven or Netbus were no big threat, but even for their limited stealth techniques they where amazingly successful in penetrating systems. Not to think of how successful they would be, when they get more sophisticated. As the scenario in Section 5 demonstrates trojan horses of the "future" will be very difficult to detect on a system. Therefore the protection tools have to evolve and adapt to this new raising threat or trojans will become widely used without a good cure against them. In the wild we already see the use increasing.

From my point of view desktop firewalls have to get a wider acceptance rate. Even thought that they on their own do not protect against sophisticated attacks, they still provide protection. If desktop firewall vendors address some of the open issues discussed in this paper, these tools will be able to add an extra level of protection to the systems. To raise consumer acceptance anti-virus vendors could bundled desktop firewalls with their products, as some already do.

Anti-Trojan tools which are needed to detect trojan horses have not yet the attention they would need to be taken seriously. Maybe some of their features will get implemented in future anti-virus tools. Because without specialized tools or at least a different approach than signature scanning, it will be very difficult to stop modern trojan horses from infiltrating networks. This is even true when not taking metamorphic code into consideration. Therefore specialized anti-trojan tools have to be developed and used by the users. To achieve this, the awareness of the trojan threat has to be increased.

We see that there is still some room for improvement for security tools, if they want to be able to counter sophisticated trojan horse attacks. If the marked realizes the threat and acknowledges that there has to be done something against it, good chances for a reasonable protection can be expected.

Bibliography

[1] Summers, Rita C. Secure Computing Threats and Safeguards, McGraw Hill, 1997.

[2] Brumley, David. “Rootkits - How Intruders Hide”
URL: http://ouah.kernsh.org/Drootkits.html (2.1.2004)

[3] Von Braun, Joakim. “Ports used by trojans”, 9.12.2003
URL: http://www.simovits.com/sve/nyhetsarkiv/1999/nyheter9902.html
(21.12.2003)

[4] Symantec, “Symantec PCAnywhere”
URL:
http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=2
(20.12.2003)

[5] Carvey, H. “The Dark Side of NTFS (Microsofts Scarlet Letter)”
URL: http://patriot.net/~carvdawg/docs/dark_side.html (11.11.2003)

[6] ZoneLabs, “ZoneAlarm Pro 4”
URL:
http://www.zonelabs.com/store/content/catalog/products/zap/zap_details.jsp
(5.1.2004)

[7] Oberhumer, Markus F.X.J. “UPX: the Ultimate Packer for eXecutables”, 7.11.2002
URL: http://upx.sourceforge.net (19.12.2003)

[8] Diamond CS, “Trojan Defence Suite”
URL: http://tds.diamondcs.com.au/ (27.12.2003)

[9] Diamond CS, “Freeware Autostart Viewing and Control Tool for Windows”
URL: http://www.diamondcs.com.au/index.php?page=asviewer (28.12.2003)

[10] Wilders, “Anti-Trojan tools”, 23.12.2003
URL: http://www.wilders.org/anti_trojans.htm (20.12.2003)

[11] Keir, Robin. “How to bypass your personal firewall outbound detection”, 25.3.2002
URL: http://keir.net/firehole.html (5.1.2004)

[12] Security Focus, “Sygate Personal Firewall DLL Authentication Bypass Vulnerability”, 29.12.2003
URL: http://www.securityfocus.com/bid/9312 (3.1.2004)

[13] Rauen, Mathias. “Madshi’s source code examples”
URL: http://www.madshi.net/ (8.1.2004)

[14] Hackbusters, “Outbound, personal firewall leak test”
URL: http://www.hackbusters.net/ob.html (22.12.2003)

[15] Rudis, Bob. “The Enemy Within: Firewalls and Backdoors”, 9.6.2003
URL: http://www.securityfocus.com/infocus/1701 (21.12.2003)

[16] Brinkhoff, Lars. ”HTTP tunnel service”, 10.1.2003
URL: http://www.nocrew.org/software/httptunnel.html (29.12.2003)

[17] Lycos, “Free website hoster Tripod”
URL: http://www.tripod.lycos.com (2.1.2004)