NETWORK SECURITY PROBLEMS

Gábor Ferenczy
Associate professor
„Miklós Zrínyi" National Defense University

A basic understanding of computer networks is requisite in order to understand the principles of network security. In this document, we'll discuss some of the threats that managers and administrators of computer networks need to confront, and then some tools that can be used to reduce the exposure to the risks of network computing. Following that, we'll take look at computer viruses.

INTRODUCTION

There are two extremes in network security: absolute security and absolute access. The closest we can get to an absolutely secure machine is one unplugged from the network, power supply, locked in a safe, and thrown at the bottom of the ocean. Unfortunately, it isn't terribly useful in this state. A machine with absolute access is extremely convenient to use: it's simply there, and will do whatever you tell it, without questions, authorization, passwords, or any other mechanism. Unfortunately, this isn't terribly practical, either: the Internet is a bad neighborhood now, and it isn't long before some bonehead will tell the computer to do something like self-destruct, after which, it isn't terribly useful to you.

Every user and organization needs to decide for itself where between the two extremes of total security and total access they need to be. A policy needs to articulate this, and then define how that will be enforced with practices and such. Everything that is done in the name of security, then, must enforce that policy uniformly.

TYPES AND SOURCES OF NETWORK THREATS

First of all, we'll get into the types of threats there are against networked computers, and then some things that can be done to protect yourself against various threats.

DENIAL-OF-SERVICE

DoS (Denial-of-Service) attacks are probably the nastiest, and most difficult to address. These are the nastiest, because they're very easy to launch, difficult (sometimes impossible) to track, and it isn't easy to refuse the requests of the attacker, without also refusing legitimate requests for service.

The premise of a DoS attack is simple: send more requests to the machine than it can handle. There are toolkits available in the underground community that make this a simple matter of running a program and telling it which host to blast with requests. The attacker's program simply makes a connection on some service port, perhaps forging the packet's header information that says where the packet came from, and then dropping the connection. If the host is able to answer 20 requests per second, and the attacker is sending 50 per second, obviously the host will be unable to service all of the attacker's requests, much less any legitimate requests (hits on the web site running there, for example).

Such attacks were fairly common in late 1996 and early 1997, but are now becoming less popular.

Some things that can be done to reduce the risk of being stung by a denial of service attack include

— Not running your visible to the world servers at a level too close to capacity
— Using packet filtering to prevent obviously forged packets from entering into your network address space.
— Keeping up to date on security related patches for your hosts' operating systems.

UNAUTHORIZED ACCESS

“Unauthorized access”' is a very high-level term that can refer to a number of different sorts of attacks. The goal of these attacks is to access some resource that your machine should not provide the attacker. For example, a host might be a web server, and should provide anyone with requested web pages. However, that host should not provide command shell access without being sure that the person making such a request is someone who should get it, such as a local administrator.

— **Executing Commands Illicitly:** It's obviously undesirable for an unknown and untrusted person to be able to execute commands on your server machines. There are two main classifications of the severity of this problem: normal user access, and administrator access. A normal user can do a number of things on a system (such as read files, mail them to other people, etc.) that an attacker should not be able to do. This might, then, be all the access that an attacker needs. On the other hand, an attacker might wish to make configuration changes to a host (perhaps changing its IP address, putting a start-up script in place to cause the machine to shut down every time it's started, or something similar). In this case, the attacker will need to gain administrator privileges on the host.

— **Confidentiality Breaches:** We need to examine the threat model: what is it that you're trying to protect yourself against? There is certain information that could be quite damaging if it fell into the hands of a competitor, an enemy, or the public. In these cases, it's possible that compromise of a normal user's account on the machine can be enough to cause damage (perhaps in the form of PR, or obtaining information that can be used against the company, etc.)

While many of the perpetrators of these sorts of break-ins are merely thrill-seekers interested in nothing more than to see a shell prompt for your computer on their screen, there are those who are more malicious, as we'll consider next. (Additionally, keep in mind that it's possible that someone who is normally interested in nothing more than the thrill could be persuaded to do more: perhaps an unscrupulous competitor is willing to hire such a person to hurt you.)

— **Destructive Behavior:** Among the destructive sorts of break-ins and attacks, there are two major categories.

- *Data Diddling.* The data diddler is likely the worst sort, since the fact of a break-in might not be immediately obvious. Perhaps he's toying with the numbers in your spreadsheets, or changing the dates in your projections and plans. Maybe he's changing the account numbers for the auto-deposit of certain paychecks. In any case, rare is the case when you'll come in to work one day, and simply know that something is wrong. An accounting procedure might turn up a discrepancy in the books three or four months after the fact. Trying to track the problem down will certainly be difficult, and once *that* problem is discovered, how can any of your numbers from that time period be trusted? How far back do you have to go before you think that your data is safe?
- *Data Destruction:* Some of those perpetrate attacks are simply twisted jerks who like to delete things. In these cases, the impact on your computing capability — and consequently your business — can

be nothing less than if a fire or other disaster caused your computing equipment to be completely destroyed.

ATTACKS AGAINST IP

The next source of network threats the protocol of Internet. *TCP/IP* (Transport Control Protocol/Internet Protocol) is the ”language” of the Internet. Anything that can learn to “speak TCP/IP” can play on the Internet. This is functionality that occurs at the Network (IP) and Transport (TCP) layers in the ISO/OSI Reference Model.

A number of attacks against IP are possible. Typically, this exploit the fact that IP does not perform a robust mechanism for *authentication*, which is proving that a packet came from where it claims it did. A packet simply claims to originate from a given address, and there isn't a way to be sure that the host that sent the packet is telling the truth. This isn't necessarily a weakness, *per se*, but it is an important point, because it means that the facility of host authentication has to be provided at a higher layer on the ISO/OSI Reference Model. Today, applications that require strong host authentication (such as cryptographic applications) do this at the application layer.

— **IP Spoofing:** This is where one host claims to have the IP address of another. Since many systems (such as router access control lists) define which packets may and which packets may not pass based on the sender's IP address, this is a useful technique to an attacker: he can send packets to a host, perhaps causing it to take some sort of action.

— Additionally, some applications allow login based on the IP address of the person making the request. These are both good examples how trusting untrustable layers can provide security that is — at best — weak.

— **IP Session Hijacking:** This is a relatively sophisticated attack and very dangerous, because there are now toolkits available in the underground community, that allow otherwise unskilled bad-guy-wannabes to perpetrate this attack. IP Session Hijacking is an attack whereby a user's session is taken over, being in the control of the attacker. If the user was in the middle of email, the attacker is looking at the email, and then can execute any commands he wishes as the attacked user. The attacked user simply sees his session dropped, and may simply login again, perhaps not even noticing that the attacker is still logged in and doing things.

— This can be solved by replacing standard telnet-type applications with encrypted versions of the same thing. In this case, the attacker can still

take over the session, but he'll see only “gibberish” because the session is encrypted. The attacker will not have the needed cryptographic key(s) to decrypt the data stream from `G`, and will, therefore, be unable to do anything with the session.

WHERE DO THEY COME FROM?

How, though, does an attacker gain access to your equipment? Through any connection that you have to the outside world. This includes Internet connections, dial-up modems, and even physical access. (How do you know that one of the temps that you've brought in to help with the data entry isn't really a system cracker looking for passwords, data phone numbers, vulnerabilities and anything else that can get him access to your equipment?)

In order to be able to adequately address security, all possible avenues of entry must be identified and evaluated. The security of that entry point must be consistent with your stated policy on acceptable risk levels.

From looking at the sorts of attacks that are common, we can divine a relatively short list of high-level practices that can help prevent security disasters, and to help control the damage in the event that preventative measures were unsuccessful in warding off an attack.

SECURE NETWORK DEVICES

FIREWALLS

As we've seen in our discussion of the Internet and similar networks, connecting an organization to the Internet provides a two-way flow of traffic. This is clearly undesirable in many organizations, as proprietary information is often displayed freely within a corporate *intranet* (that is, a TCP/IP network, modeled after the Internet that only works within the organization).

In order to provide some level of separation between an organization's intranet and the Internet, *firewalls* have been employed. A firewall is simply a group of components that collectively form a barrier between two networks.

Types of Firewalls

There are three basic types of firewalls, and we'll consider each of them:

— **Application Gateways:** The first firewalls were application gateways, and are sometimes known as proxy gateways. These are made up of bastion hosts that run special software to act as a proxy server. This software runs at the *Application Layer* of the ISO/OSI Reference Model, hence the name. Clients behind the firewall must be *proxitized* (that is, must know how to use the proxy, and be configured to do so) in order to use Internet services. Traditionally, these have been the most secure, because they don't allow anything to pass by default, but need to have the programs written and turned on in order to begin passing traffic.

— These are also typically the slowest, because more processes need to be started in order to have a request serviced.

— **Packet Filtering:** Packet filtering is a technique whereby routers have *ACLs* (Access Control Lists) turned on. By default, a router will pass all traffic sent it, and will do so without any sort of restrictions. Employing ACLs is a method for enforcing your security policy with regard to what sorts of access you allow the outside world to have to your internal network, and vice versa.

— There is less overhead in packet filtering than with an application gateway, because the feature of access control is performed at a lower ISO/OSI layer (typically, the transport or session layer). Due to the lower overhead and the fact that packet filtering is done with routers, which are specialized computers optimized for tasks related to networking, a packet filtering gateway is often much faster than its application layer cousins.

— Because we're working at a lower level, supporting new applications either comes automatically, or is a simple matter of allowing a specific packet type to pass through the gateway. (Not that the *possibility* of something automatically makes it a good idea; opening things up this way might very well compromise your level of security below what your policy allows.)

— There are problems with this method, though. Remember, TCP/IP has absolutely no means of guaranteeing that the source address is really what it claims to be. As a result, we have to use layers of packet filters in order to localize the traffic. We can't get all the way down to the actual host, but with two layers of packet filters, we can differentiate between a packet that came from the Internet and one that came from our internal

network. We can identify which network the packet came from with certainty, but we can't get more specific than that.

— **Hybrid Systems:** In an attempt to marry the security of the application layer gateways with the flexibility and speed of packet filtering, some vendors have created systems that use the principles of both.

— In some of these systems, new connections must be authenticated and approved at the application layer. Once this has been done, the remainder of the connection is passed down to the session layer, where packet filters watch the connection to ensure that only packets that are part of an ongoing (already authenticated and approved) conversation are being passed.

— Other possibilities include using both packet filtering and application layer proxies. The benefits here include providing a measure of protection against your machines that provide services to the Internet (such as a public web server), as well as provide the security of an application layer gateway to the internal network. Additionally, using this method, an attacker, in order to get to services on the internal network, will have to break through the access router, the bastion host, and the choke router.

SECURE MODEMS; DIAL-BACK SYSTEMS

It's important to remember that the firewall only one entry point to your network. Modems, if you allow them to answer incoming calls, can provide an easy means for an attacker to sneak *around* (rather than *through*) your front door (or, firewall). Just as castles weren't built with moats only in the front, your network needs to be protected at all of its entry points. If modem access is to be provided, this should be guarded carefully. The *terminal server* , or network device that provides dial-up access to your network needs to be actively administered, and its logs need to be examined for strange behavior. Its pasword need to be strong — not ones that can be guessed. Accounts that aren't actively used should be disabled. In short, it's the easiest way to get into your network from remote: guard it carefully.

There are some remote access systems that have the feature of a two-part procedure to establish a connection. The first part is the remote user dialing into the system, and providing the correct userid and password. The system will then drop the connection, and call the authenticated user back at a known telephone number. Once the remote user's system answers that call, the connection is established, and the user is on the network. This works well for folks working at home, but can be problematic for users wishing to dial in from hotel rooms and such when on business trips.

Other possibilities include one-time password schemes, where the user enters his userid, and is presented with a challenge, a string of between six and eight numbers. He types this challenge into a small device that he carries with him that looks like a calculator. He then presses enter, and a ``response'' is displayed on the LCD screen. The user types the response, and if all is correct, he login will proceed. These are useful devices for solving the problem of good passwords, without requiring dial-back access. However, these have their own problems, as they require the user to carry them, and they must be tracked, much like building and office keys.

No doubt many other schemes exist. Take a look at your options, and find out how what the vendors have to offer will help you *enforce your security policy effectively*.

CRYPTO-CAPABLE ROUTERS

A feature that is being built into some routers is the ability to session encryption between specified routers. Because traffic traveling across the Internet can be seen by people in the middle who have the resources (and time) to snoop around, these are advantageous for providing connectivity between two sites, such that there can be secure routes.

VIRTUAL PRIVATE NETWORKS

Given the ubiquity of the Internet, and the considerable expense in private leased lines, many organizations have been building *VPNs* (Virtual Private Networks). Traditionally, for an organization to provide connectivity between a main office and a satellite one, an expensive data line had to be leased in order to provide direct connectivity between the two offices. Now, a solution that is often more economical is to provide both offices connectivity to the Internet. Then, using the Internet as the medium, the two offices can communicate.

The danger in doing this, of course, is that there is no privacy on this channel, and it's difficult to provide the other office access to ``internal'' resources without providing those resources to everyone on the Internet.

VPNs provide the ability for two offices to communicate with each other in such a way that it looks like they're directly connected over a private leased line. The session between them, although going over the Internet, is private (because the link is encrypted), and the link is convenient, because each can see each others' internal resources without showing them off to the entire world.

A number of firewall vendors are including the ability to build VPNs in their offerings, either directly with their base product, or as an add-on. If you have need to connect several offices together, this might very well be the best way to do it.

THE VIRUSES

Not only the network attacks mean risk for us, but the computer viruses also. What is a virus? Virus is a parasitic program written intentionally to enter a computer without the user's permission or knowledge. The word parasitic is used because a virus attaches to files or boot sectors and replicates itself thus continuing to spread. Though some viruses do little but replicate, others can cause serious damage or affect program and system performance. A virus should never be assumed harmless and left on a system.

Viruses, worms, Trojan Horses, and logic bombs are all unwanted, uninvited, potentially dangerous software, but there are important distinctions among them. The differences lie in whether the category requires a host program and *whether* it makes copies of itself. All four may cause damage, but this is not integral to the definitions. The table defines each one

TYPES OF VIRUSES

Virus are classified by the ways they infect computer systems:

- **Program:** Executable program files such as .Com, .Exe, .Ovl, .Drv, .Sys, .Bin
- **Boot:** Boot Record, Master Boot, FAT and Partition Table.
- **Multipartite:** Both program and boot infector.

HOW VIRUSES CONTAMINATE AND SPREAD?

A virus is inactive until the infected program is run or boot record is read. As the virus is activated it loads into the computers memory where it can perform a triggered event or spread itself. Disks used in an infected system can then carry the virus to another machine. Programs downloaded from bulletin boards can also spread a virus. Data files, however, can not transfer a virus but they can become damaged.

— **Boot Infectors:** Every disk contains a boot sector whether it is a bootable disk or not. When the computer is powering up looking for the Boot information and reads an infected disk in the A: drive the virus is transfer to the computers hard drive. Once the boot code on the drive is infected the virus will be loaded into memory on every startup. From memory the boot virus can travel to every disk that is read and the infection spreads. Most Boot virus's could be on a system for a long time without causing problems. However there are some nasty ones that will destroy the boot information or force a complete format of the hard drive.
— **Program Infectors:** When an infected application is run the virus activates and is loaded into memory. While the virus is in memory any program file subsequently run becomes infected. Multiple infections are very common and will certainly cause system problems. Program files may function without any problems for some time but eventually programs have problems or multiple infection brings the system down. The data the program produces may be a first sign of infection such as saving files without proper DOS names.

VIRUS CHARACTERISTICS

Viruses normally have multiple characteristics. Their characteristics are:

— **Memory Resident:** Loads much like a TSR staying in memory where it can easily replicate itself into programs of boot sectors. Most common.
— **Non-Resident:** Does not stay in memory after the host program is closed, thus can only infect while the program is open. Not as common.
— **Stealth:** The ability to hide from detection and repair manifests in two ways.
  - Full — Virus redirects disk reads to avoid detection.
  - Size — Disk directory data is altered to hide the additional bytes of the virus.
— **Encrypting:** Technique of hiding by transformation. Virus code converts itself into cryptic symbols. However, in order to launch (execute) and spread the virus must decrypt and can then be detected.
— **Polymorphic:** Ability to mutate by changing code segments to look different from one infection to another. This type of virus is a challenge for ant-virus detection methods.
— **Triggered Event:** An action built into a virus that is set off by the date, a particular keyboard action or DOS function. It could be as simple as a message printed to the screen or serious as in reformatting the hard drive or deleting files.

— **In the Wild:** A virus is referred to as "in the wild" if is has been verified by groups that track virus infections to have caused an infection outside a laboratory situation. A virus that has never been seen in a real world situation is not in the wild, and sometimes referred to as "in the zoo".
— **Macro virus:** Macro viruses are much like other viruses in many ways: they consist of code written in such a way that under some condition, that code "reproduces", making a copy of itself. Like other viruses, they can be written to cause damage, display a message, or do anything else a program can be made to do. Macro viruses are much like other viruses in many ways: they consist of code written in such a way that under some condition, that code "reproduces", making a copy of itself. Like other viruses, they can be written to cause damage, display a message, or do anything else a program can be made to do. There are some differences between a macro virus and other kinds of viruses: Boot viruses are always written in assembly language; viruses which infect executable programs are usually written in assembly language, but sometimes in a high-level language such as C. Macro viruses are always written in a macro language. To get a boot virus, you must boot your machine with a diskette that is infected with the boot virus. To get a file virus, you need to run a copy of the infected file. To get a macro virus, all you need to do is double-click on an infected document, to view it. When it loads, its macros run, and you are infected.
— **Companion virus:** A companion virus is one which, instead of modifying an existing file, creates a new program which (unknown to the user) gets executed by the command-line interpreter instead of the intended program. (On exit, the new program executes the original program so that things will appear normal.) The only way this has been done so far is by creating an infected .COM file with the same name as an existing .EXE file. Note that those integrity checkers which look only for modifications in existing files will fail to detect such viruses.

HOW CAN WE REDUCE THE DISASTER?

— *We must have backups.* This isn't just a good idea from a security point of view. Operational requirements should dictate the backup policy, and this should be closely coordinated with a disaster recovery plan, such that if an airplane crashes into your building one night, you'll be able to carry on your business from another location. Similarly, these can be useful in recovering

your data in the event of an electronic disaster: a hardware failure, or a breaking that changes or otherwise damages your data.

— *Don't put data where it doesn't need to be.* Although this *should* go without saying, this doesn't occur to lots of folks. As a result, information that doesn't need to be accessible from the outside world sometimes is, and this can needlessly increase the severity of a break-in dramatically.
— *Avoid systems with single points of failure*. Any security system that can be broken by breaking through any one component isn't really very strong. In security, a degree of redundancy is good, and can help you protect your organization from a minor security breach becoming a catastrophe.
— *Stay current with relevant operating system patches*. Be sure that someone who knows what you've got is watching the vendors' security advisories. Exploiting old bugs is still one of the most common (and most effective!) means of breaking into systems.
— *Watch for relevant security advisories*. In addition to watching what the vendors are saying, keep a close watch on groups. Make sure that at least one person (preferably more) is subscribed to these mailing lists
— *Have someone on staff be familiar with security practices.* Having at least one person who is charged with keeping abreast of security developments is a good idea. This need not be a technical wizard, but could be someone who is simply able to read advisories issued by various incident response teams, and keep track of various problems that arise. Such a person would then be a wise one to consult with on security related issues, as he'll be the one who knows if web server software version such-and-such has any known problems, etc.
— *Make the floppy disks write-protected.* So a virus can’t infect the disks.
— *Have one or more anti/virus tools.* These programs should upgrade so often it is possible.
— *Scan all new files for viruses.*

CONCLUSIONS

Security is a very difficult topic. Everyone has a different idea of what security is, and what levels of risk are acceptable. The key for building a secure network is to define what security means to your organization. Once that has been defined, everything that goes on with the network can be evaluated with respect to that policy. Projects and systems can then be broken down into their compo-

nents, and it becomes much simpler to decide whether what is proposed will conflict with your security policies and practices.

Many people pay great amounts of lip service to security, but do not want to be bothered with it when it gets in their way. It's important to build systems and networks in such a way that the user is not constantly reminded of the security system around him. Users who find security policies and systems too restrictive will find ways around them. It's important to get their feedback to understand what can be improved, and it's important to let them know *why* what's been done has been, the sorts of risks that are deemed unacceptable, and what has been done to minimize the organization's exposure to them.

Security is everybody's business, and only with everyone's cooperation, an intelligent policy, and consistent practices, will it be achievable.

REFERENCES

[1] R.T. Morris: *A Weakness in the 4.2BSD Unix TCP/IP Software.* Computing Science Technical Report No. 117, AT&T Bell Laboratories, Murray Hill, New Jersey, 1985

[2] Y. Rekhter, R. Moskowitz, D, Karrenberg, G. de Groot, E Lear: *Address Allocation for Private Internets* RFC 1918

[3] J.P. Holbrook, J.K: *Reynolds: Site Security Handbook.* RFC 1244

[4] Matt Curtin: *Introduction to Network Security*, http://www.interhack.net/pubs/network-security/network-security.html

[5] Virus FAQ: http://www.cai.com/virusinfo/faq.htm