A Distributed Approach to Network Security

Joe Walko

Mercyhurst College

Computer Systems Department

1/15/99

May 2001

Introduction

On December 7th of last year, two programs were published on Packetstorm[1] which changed the face network security. These utilities allowed a systematic, coordinated, denial of service attack to be launched by thousands of systems simultaneously. Many web servers crumbled when hit by the flood of over 20 billion packets/second generated by the attack. However, the most amazing thing about these tools is that they were automated. The attacks came from thousands of locations around the planet, with absolutely no one behind them, and no one to be held accountable. Such is the power of distributed technology.

Firewalls alone can no longer withstand the increasingly sophiticated and organized attacks we see on a daily basis. What measures are available to counter these attacks? What are their weaknesses? How can they be improved on? These questions are considered in this distributed approach to network security. The paper assumes a basic understanding of network protocols.

What is distributed network security?

Distributed systems are not a new concept. Distributed computing has been around since the 70's, when networks became increasingly integrated[2]. Distributed security is however, a concept in its infancy. Only within the past few years have vendors such as Internet Security Systems[3] stepped forward to create network security

---

[1] Packetstorm.securify.com is a popular network security site. The programs were TribeFloodNetwork and trinoo.
[2] *Distributed Computing Environments*, Cerutti and Pierson, Ch 1.
[3] See http://xforce.iss.net/alerts/advise40.php3

systems that span entire networks. This technology has come to be known as Intrusion Detection Systems, or IDS.

IDS works on either a host-based system, network-based system, or both. The most common is the network-based IDS. The classic example of this is akin to that of a glorified packet sniffer. A machine or series of machines observe every packet on the network. Each packet is analyzed according to the available exploit "signatures" of the IDS. For example, there are many cgi-bin exploits available. An IDS system might watch for repeated attempts to access the cgi directory, and automatically mail the administrator with the attacker's IP address or other details. Host-based systems simply observe suspicious activity on various machines, and respond accordingly.

Note that distributed network security is not synonymous with IDS. IDS is just the biggest and most common type of distributed network security, and will be considered here in detail.

Conventional IDS

Benefits

The main benefit of IDS is its manageability. IDS allows centralized, large-scale ease-of-use for many different networks. For example, ISS's RealSecure 3.0 system allows administrators to easily access the status of many machines (and in turn the networks they monitor) by simply clicking. This is a breath of fresh air when compared to the sheer hell of maintaining the security of hundreds of UNIX systems across a complex multi-protocol network. Another benefit of IDS is its automation. As in the

example above, when an event is triggered different actions can be configured to run, such as email or paging various personnel.

Shortcomings

IDS is by no means the ultimate solution, as it has many significant shortcomings. Since IDS relies on exploit signatures, it is imperative for the system to have the most update collection of exploits available. This is near impossible since new vulnerabilities are found on a daily basis. As stated in *Network Computing*, "No IDS vendor has any sort of push technology in place to update its signature database."[4] This failure to recognize the newest attacks must be corrected for IDS to truly become a standard.

Since IDS must survey each packet on the wire, speed becomes an important issue. On a 10Mbps Ethernet segment, IDS will work just fine. But when placed on a network backbone running at 100Mbps all modern IDS stumble to keep up with the traffic. With new fiber being laid at higher speeds each day, IDS must constantly play catch-up.

IDS also currently lacks the ability to detect network and user trends. For example, there is a private military security system able to detect portscans as infrequent as 2 a day. IDS is not at all intelligent in this respect, and often fails to detect even obvious network traffic patterns.

Improving IDS

Here are my suggestions for improving current IDS systems:

Obtaining Timely Exploit Signatures

This is by far the biggest failure of current IDS systems. On the following page observe Figure 1. Here is an automated system for receiving new exploits, testing these exploits, and applying the needed security patches.

We begin at the new exploit database. This could be an in-house proprietary database system, or produced by a crawler program that scans already existing databases (i.e. securityfocus.com) for new vulnerabilities. Most likely you would want a proprietary database setup to ensure smooth interoperability at the later steps.

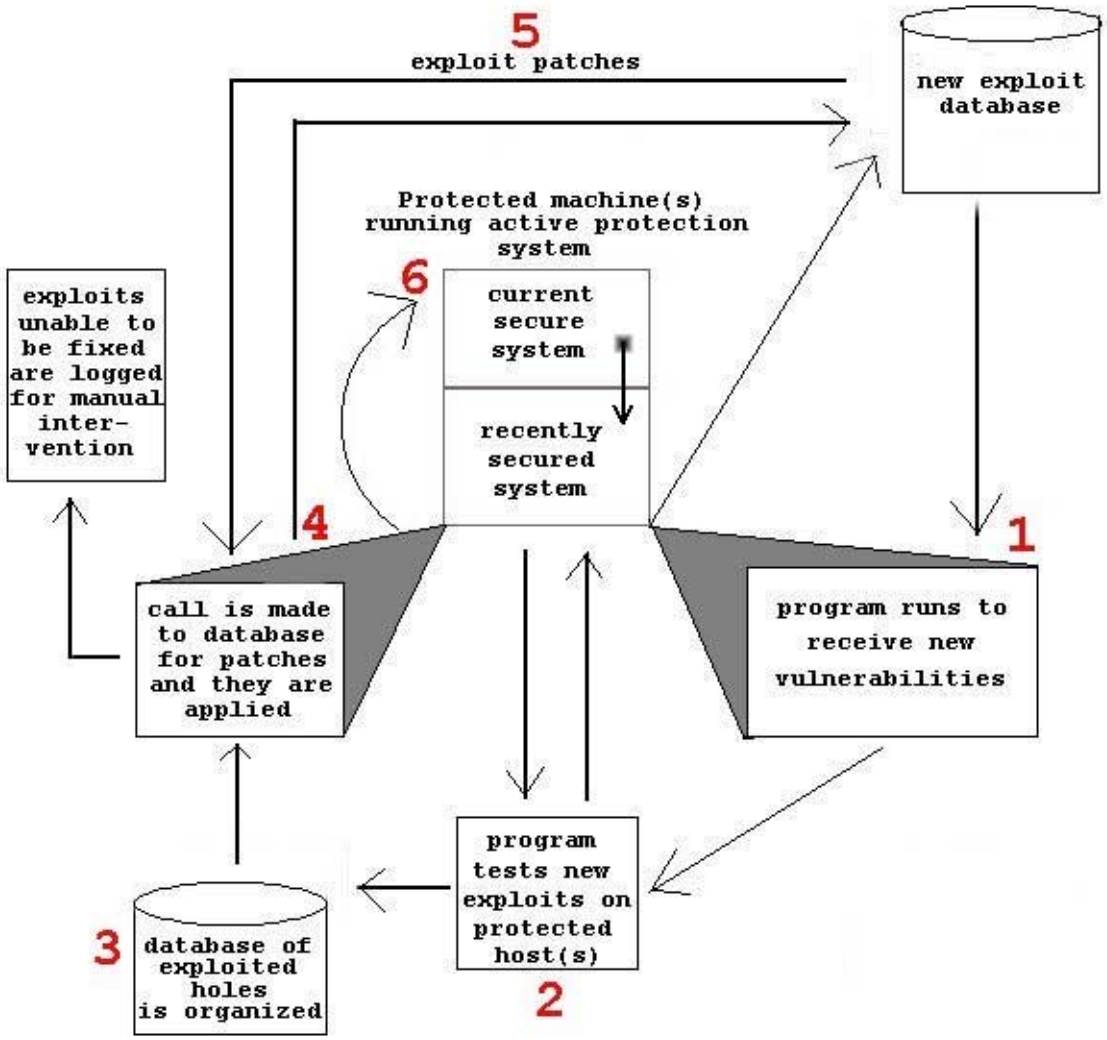
After the program receives the new exploits (step 1), penetration testing is performed on the protected host(s) (step 2). Realize that this is a truly distributed system, and each program or parts of these programs may run on various machines.

Upon completing the vulnerability checks, the program organizes another database of the exploits that were tested with success (step 3). A call is made by the existing (old) system to the new exploit database in order to receive the necessary patches (step 4). The new exploit database machine then sends the patches to the running program (step 5). The patches are then applied and a new (secure) system exists. The patches that could not be supplied are logged for later human intervention. Eventually the current secure system will become vulnerable again, and the cycle will repeat. I refer to this scheme as the distributed active protection system.

---

[4] *Network Computing*, November 15, 1999, page 96.

Figure 1



Speed Issues

The best way to eliminate the stress on a network-based IDS station is to designate the various stations to only sniff specific packets. In this way the traffic could be equally divided over multiple stations. For instance, one station monitors all inbound TCP traffic on the local subnet, while another station monitors all outbound NetBIOS traffic to another network. This would lighten the load significantly without compromising security for an only slightly higher cost.

Offensive Security

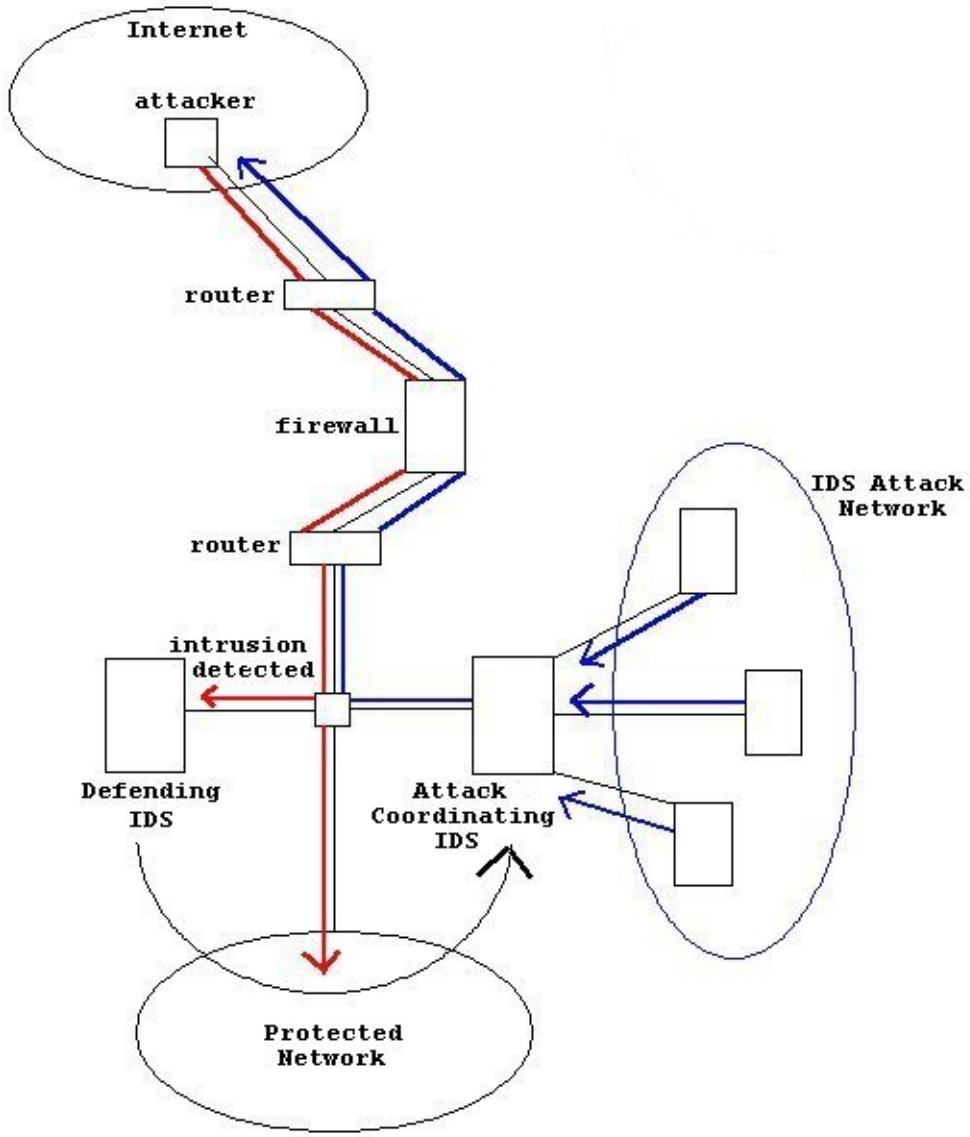
IDS is primarily a defensive concept by nature. The system waits to be attacked and then responds rather passively. There are a few firewall vendors who employ schemes to modify routing tables (to deny packets from an offending domain) when an attack is recognized, or masquerade as a totally different machine[5].

IDS's distributed nature naturally lends itself to something that has been in the UNIX community for ages: offensive security. Instead of simply hiding from, logging, or denying access when attacked, IDS should be designed to *counterattack*. Figure 2 on the following page illustrates this method.

As you see, the attacker bypasses the firewall and routers, and gains access to the intranet. An IDS machine (Defending IDS) recognizes this and relay's the attacker's whereabouts to an IDS attack-coordinating machine. The job of this IDS machine is to organize the counterattack by recruiting other machines on the intranetwork (or internet for that matter) to launch various denial of service attacks against the offending host (similar to Trinoo of TFN). Most likely, the IDS team will win. This is because your average attacker does not have bandwidth (though he can launch a pretty hefty attack from multiple hosts) comparable to a large corporate or military network attached to the Internet. The attacker's machine will most likely crumble under the pressure[6].

[5] I once heard of a man who thought he had gained root access on a phone company's Unix box, only to discover later he was caught in one of the seven fake security subsystems. These fake systems were busy all night logging his actions – all without any threat to the real data.

Figure 2



This counterattacking strategy would be most useful in cyberwarfare. For example, during the attack on Serbia last year, the US made an effort to cripple the military's communications infrastructure by attacking their computer systems[7]. Had the Serbian's had offensive security measures in place as well as defensive ones, they might

[6] Note however, that the IDS attackers don't simply have to launch a flood attack. Many other methods of attack could be employed here.
[7] See http://slashdot.org/articles/99/10/08/1226209.shtml

have fared much better, especially considering the weak security measures in place on many US military machines.

Added Security

One idea to improve the security features of IDS is something I refer to as intelligent path analysis. This is done by evaluating a packet's information (routing information in headers, protocol details etc.) based on a specific criterion to determine if it is an authorized packet. For example, a seemingly valid packet can be deemed suspicious if it arrives from an unlikely origin or by an unlikely protocol. An example of this would be an IPX packet coming from a subnet where no IPX-based applications exist. This programming is more suited to a company's intranet, as suspicious datagrams coming from external sources would most likely be firewalled immediately. I believe this would greatly improve the ability of IDS to recognize potential threats.

Defeating Current IDS

Currently, there are many programs and methods by which to bypass modern-day IDS. Nmap[8], a popular network scanner, is easily able to accomplish this. As stated in *Network Computing*: "By playing with timing thresholds and scanning methods, we could frequently avoid detection while still gaining valuable reconnaissance data."[9] As a general rule, IDS systems usually only detect very "loud" scans. That is, scans that do not conceal themselves with randomization or timing mechanisms.

[8] See its homepage at www.insecure.org
[9] *Network Computing*, May 17, 1999

Spoofing attacks[10] also usually bypass IDS. Current IDS does not use tactics such as router path tracing or intelligent path analysis, so an IDS system usually believes anything it sees on the wire. Spoofing has long been a problem and will likely be so in the future.

Standard host-based IDS systems, such as the time-tested Tripwire for UNIX, are usually easy to avoid. For instance, if the intruder knows much about the IDS system in question, he would know what logs to avoid writing to, what commands not to run, etc. So when he obtains an account on the system, he will stick to legitimate uses of the account to leverage further access. This could be accomplished by convincing another user to run a shell script with subversive functions (trojan horses).

Defeating Future IDS

Security that goes untested is worse than no security at all. Here I present ideas and a tool for defeating the future of Internet security.

The best way to attack distributed security measures is to fight distribution with distribution. An example of this would be slow, random scans from multiple locations, with mixes of legitimate or logical packets (i.e. ARP requests) mixed in (Nmap can do this). Recognizing the attack signals amidst the extra network noise is near impossible on highly congested networks.

Viruses and trojan horses will continue to be a threat to the most advanced of systems. This is because new viruses are discovered every day, and most vendors only provide updates on a monthly basis. If a user unknowingly installs a malicious program

[10] Spoofing is a method by which the attacker creates false IP header information to trick a host into believing he is a different machine. For a good explanation see *Hacker Proof: The Ultimate Guide to Network Security*, by Lars Klander, Edward J., and Jr. Renehan.

designed to connect to an external machine, no IDS system no matter how up-to-date will be able to recognize the threat.

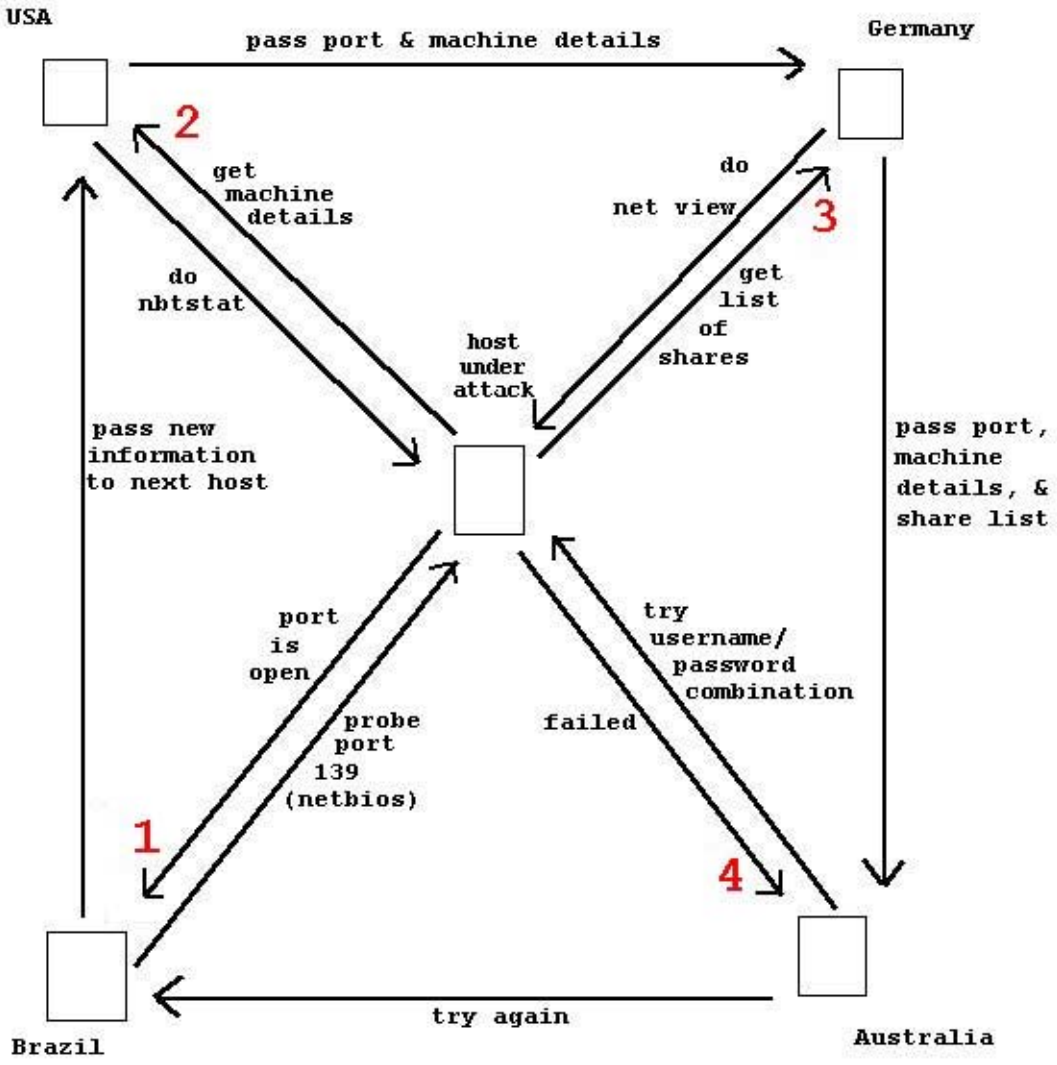
I believe network distractions and "decoy threats" will play a large role in bypassing IDS. A decoy threat is designed to trip an alarm for one reason or another, and while the proverbial sirens are sounding, do the real damage. For example, say an attacker wanted to attempt various username/password combinations on a host. This might normally set off IDS alarms. But just prior to this, the attacker spoofed his IP and ran an exploit scanner on a range of other hosts. The login attempts may still be reported by the IDS, but chances are they will be overlooked by the administrators or even the IDS itself.

Distributed Attack System

On the following page observe Figure 3. Here we see an attack system unlike anything previously conceived. It is a globally distributed, self-updating, intelligent, stealth vulnerability/port scanner. Each network attacker runs both a client and a server program. When an attacker probes a host and obtains information, this information is sent by its client to the next attacker's server. Information is built up over time allowing for more intelligent and effective attacks.

Let us consider the situation in Figure 3. The first attacker, located in Brazil, probes port 139 and finds it is open. It passes this information along to the next attacker in the US. The attacker in the US, now knowing that port 139 is open, performs an nbtstat[11] to obtain the details of the machine. The US attacker now passes the port and the machine details to an

Figure 3



attacker in Germany. The German machine, now knowing all the above information, does a net view[12] to obtain the list of shares. The German attacker then passes all this information to the Australian attacker, who, now knowing the list of shares, performs the first login attempt. This particular username/password combination fails, and this result is sent to the next

[11] Nbtstat is a DOS utility used to gain information about another Windows computer. Try it from your own DOS prompt. Just type: "nbtstat –A x.x.x.x" without the quotations, where the Xs is the IP address of the computer.

[12] Try "net view \\x.x.x.x" for your DOS prompt.

attacker to try a different combination. This sequence of events could happen over the course of a series of minutes, hours, or days, depending on the amount of stealth required.

The implications of this form of attack are staggering. First of all, there can be any number of attackers. Here we see four, but may as well be four thousand. The more attackers, the more probes, the more intelligent the attacks based on the information gathered. Secondly, due to its thoroughly distributed nature, exactly *who* can be held accountable? Is it the person who started the scan? Everyone? Here we are talking about a program that will be freely available and in mass use. If this program is indeed created, millions will download and use it. Third, since these attacks are coming from all over the planet, there is no particular pattern. For example, there would be thousands of scan "rings", all depending on who the attacker chooses to pass the information off to next. Firewalls and routers could not deny all these domains, there would be too many. And IDS could never decipher an isolated attack from a systematic scan. Within a short time, this may become the next big distributed attack tool.

This tool would be extremely valuable with issues in national security. In peacetime, it would be a great asset in determining the status of a country's network security structure. In wartime, it would be an invaluable tool for launching large-scale, intelligent attacks on a nation's computer network.

Summary

In conclusion, distributed technology is the cutting edge of network security. Whether used to attack or defend, its uses are only limited by the resources available. As

these resources (bandwidth or memory) grow by leaps and bounds, so do the opportunities available in distributed network security.

Bibliography

Building Internet Firewalls, D. Brent Chapman, et al, O'Reilly & Associates, Published 1995

Cisco Security Architectures, Gilbert Held, Kent Hundley, McGraw-Hill, Published 1999

Distributed Computing Environments, Cerutti and Pierson, McGraw-Hill, 1993

Distributed Intruder Systems
http://ciac.llnl.gov/ciac/papers/Distributed_System_Intruder.html

Designing Network Security, Merike Kaeo, Cisco Press, Published 1999

Firewalls and Internet Security: Repelling the Wily Hacker, William R. Cheswick, Steven M. Bellovin, Addison-Wesley Publishing Company, Published 1994

Hacking Exposed: Network Security Secrets and Solutions, Stuart McClure, et al, McGraw-Hill, Published 1999

Hacker Proof: The Ultimate Guide to Network Security, Lars Klander, Edward J., Jr. Renehan, Jamsa Press, Published 1997

National Infrastructure Protection Center
http://www.fbi.gov/nipc/trinoo.htm

Network Computing, November 15-17, 1999

All graphics by Joe Walko