Disarming offense to facilitate defense[*]

D. Bruschi, E. Rosti
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
Via Comelico 39, 20135 Milano – Italy
{*bruschi, rose*}*@dsi.unimi.it*

Abstract

Computer security has traditionally focused on system defense, concentrating on victim machines protection and recovery. Moving from the opposite perspective, we propose a matching approach that focuses on limiting the attacking capabilities of the hosts. Software design and implementation weaknesses usually are at the basis of computer offensive capacities. Since software redesign or patching on an extensive basis is not possible, we propose the introduction of a filtering strategy to block abuse attempts at the originating machines. As an example, applications of such an approach are presented at network level, in order to prevent popular DoS attacks, among others.

The proposed solution is not a "panacea" and could be bypassed by sophisticated users. However, we believe it can effectively restrain the offensive capabilities of hosts that could be easily seized by crackers. We discuss the pros and cons of the proposed solution and present an application to network security.

1 Introduction

Since its origins in the early '60, computer security has focused on system defense and protection of victim machines. A variety of tools and methodologies have been proposed, many of which proved to be quite effective in protecting systems and networks from intruders. In the mid to late '80, the computing paradigm started shifting from the host to the network and became a full scale reality in the early '90. The focus of computer security should have shifted too. In the networked environment a "pacifist" host can suddenly and, sometimes, involuntarily become an attacker, that is, a threat for the entire community.

[*]This work was partially supported by the Italian M.U.R.S.T. 60% project.

Another consequence is the exacerbation of two security related problems:

- liability issues: in several countries computer owners are liable for all the actions executed by their systems. This means that they can be legally prosecuted when attacks to another system are launched from their computers, although intruders gained unauthorized access to them and are responsible for the attack;
- sophisticated intrusion tools: a clever exploitation of the network centered computing paradigm with the realization that the "network is the computer" has lead to the development of distributed tools for intrusion, which recently showed to be quite effective in launching DoS attacks at high profile Web sites.

The common answer to these problems offered by computer security is "improve your protections." On the other hand, it is becoming harder and harder for nowadays typical protections, such as firewalls and IDS's, to keep up with the ever increasing speed of network components because processors cannot process packets fast enough.

In this paper we propose a new approach to address security problems. We start by observing that a computer may as well be a victim and an attacker. Thus, if we want to reduce security threats, we should not only protect our systems but also prevent them from doing any harm. Based on this observation, we propose a new research direction for computer security whose main goal is the definition of new techniques and methodologies for building non-offending, or *disarmed* computers. In our model, *a disarmed host is a host equipped with tools that turn off the host attacking capabilities and that force the host to be reinstalled for it to be subverted.*
The tools that turn off the offending capabilities can be thought of as filters that monitor the host activity and block it when it does not conform to a "good"

behavior or, vice-versa, when it matches an "anomalous" behavior, depending on the approach followed. Such filters can be thought of as intrusion inhibitors and behave like intrusion detection systems on the attacking host.

In our perspective, attacks can be divided into two classes: attacks that can be prevented at the source and attacks that can be blocked at the destination. The latter are well known to the security community since they have been among the major subjects of computer security studies. Their characterization has lead to the definition of signatures databases for intrusion detection systems. On the contrary, little if any use has been done of attack characterization at the source in order to block offending activities as they are being performed at the source. Although the two classes of attacks have a large intersection, they are different. As an example, attacks that use IP spoofing are easier to detect at the source but can hardly ever be detected at the destination, even if heuristics such as DNS reverse lookup may be adopted to discover the spoofing in most cases.

We believe that the disarming technology can be easily adopted in local environments. In this case, its deployment can provide an effective solution to problems such as:

- liability: from a legal point of view, a disarmed computer could be considered adequately configured to comply with the law imposing that hosts not be attack sources, thus relieving the owner from liability issues;

- intranet security: disarming filters can protect an intranet from insiders' attacks and can help preventing insiders from using the internal hosts to attack computers outside the intranet perimeter. They are transparent to final users and applications, thus they do not require application customization, as it is the case with well known access control systems such as Kerberos [10].

The large scale deployment of a disarming technology would represent a further step towards a definitive answer to the following problems:

- security tools performance: because a fair share of attacks would be blocked at their source, thus never reaching the destination, tools such as firewalls and IDS's would have far less suspicious traffic to control, which translates into fewer cases to test, thus improving their operational speed [9];

- distributed tools for intrusion: with our approach, the network remains the "new computer" but not for intruders, who would have difficulties in finding hosts where their agents for distributed attacks could be installed.

Successfully deploying such an approach, however, on a geographic scale is not an easy task but we believe it to be a reasonable one. Although this kind of disarming filters could be bypassed by sophisticated users like any software protection, they could be an effective protection against abuses by the so called "script kiddies" that use ready made attack programs down-loaded from the Internet (e.g., Teardrop, Land, Smurf). Since we will implement them as kernel modules, an intruder who wants to bypass them would have to install a stripped version of the operating system, which may not be so immediate to do for unexperienced users. Furthermore, in order to be able to use victim computers, which were successfully compromised, to launch attacks, the OS should be reinstalled, which is quite conspicuous a task to perform to go unnoticed. A hardware implementation based on ASIC technology could be adopted as encouraging results will be obtained and the approach further refined.

As an example, in this paper we describe the design of a tool that can be used to block several well known attacks at the source, i.e., disarm the hosts with respect to the considered attacks. We propose a solution for blocking popular Denial of Service attacks. It requires that a set of functionalities be added to a kernel device driver in order to detect harmful packets characterizing an attack in the outgoing flow.

This paper is organized as follows. Network filters are discussed in Section 2. Section 4 discusses the proposed approach, outlines directions of future research and concludes the paper.

2 Application to Network Attacks

In this section we describe the design of a disarming filter against network attacks. Filtering components may be added as middleware between the device drivers and the kernel, so that they can monitor all the outgoing traffic, without changing the existing system. They can be executed on host computers as well as network components such as routers. The packet flow is checked against attack signatures and blocked when an attack attempt is detected, similarly to what an IDS would do on the incoming traffic. The filters rely on a signature database that stores the attack characterizations, i.e., the behavioral patterns typical of the various attacks. The more unique the attack pattern behavior, the more precise the action of the filters, i.e., the less false negative and false positive signals the filters will send. A separate module that handles critical situations, e.g., by blocking the suspicious traffic, raising alarm, suspending the allegedly offending program, or sending messages to the superuser according to some defined policy, is signaled by the filter whenever a tentative attack is detected.

Among the most (in)famous and disruptive network attacks are Denial of Service attacks such as SYN flood [2], Smurf [7], Ping of Death [3], Land [4], Teardrop [4]. Blocking this type of attacks at the target is expensive and resource consuming, both in terms of network bandwidth and CPU time. If they could be blocked at the source, intrusion detection systems could be relieved of a significant burden and focus on new attack types.

The common feature of all these attacks is the lack of strong authentication of the source address in IP packets that allows forged source addresses to be used. It allows the crackers to protect their identity and often also damage an unaware indirect victim. Additionally, each of these attacks has a distinctive behavior. At the basis of the SYN flood attack is the limited backlog of uncompleted connections allowed during the establishment of a TCP connection when the three way handshake protocol is executed. The unrestrained use of the broadcast address is at the basis of the Smurf attack. The possibility to send oversized control packets is at the basis of the Ping of Death attack. The possibility of spoofing the $< host, port >$ source address pair and setting it equal to the $< host, port >$ destination address pair thus leading the victim host to a possibly lethal loop, is at the basis of the Land attack. The need to fragment and re-assembly packets exceeding the minimum MTU of the intermediate networks traversed along the route from source to destination is at the basis of the Teardrop attack. We illustrate here how the middleware approach we propose can be employed to prevent a machine from launching some of these attacks or at least to mitigate their severity and impact on the target machine.

2.1 Source Address Spoofing

While verifying the authenticity of a packet source address at the destination is quite difficult, it is very easy to do it at the source itself. The filter we propose can apply the simple filtering rule [5] that prevents packets with a source address different from the one of the local machine to be passed to the network. Only packets carrying the proper source address, i.e., the one of the machine actually sending the packet, are allowed to the network card[1].

This simple rule is very strict and could limit network management activities, although it is sufficient to turn off most denial of service attacks as they usually forge packets with spoofed IP source addresses. Ad hoc less strict filtering rules can be adopted that verify the simultaneous presence of a spoofed IP source address and other attack specific conditions.

The simple but dangerous attack known as Land can crash or hang the victim machine by sending it packets with the same $< host, port >$ pair in the source and destination address fields. The ad hoc rule in this case would check for packets with the same destination and source address pairs.

The Smurf attack also could not be performed if spoofed addresses were not allowed, or the attacker would hang his/her network. In this case, spoofing is combined with the abuse of the broadcast address of a network, i.e., address 255, in order to flood two networks. A conspicuous traffic of ICMP ECHO_REQUEST packets is sent to the IP broadcast address of a large network (the amplifier) with spoofed source addresses of another network (the victim). If the ECHO_REQUEST packets are delivered, most receiving hosts will reply to the victim, thus flooding the alleged source network with ICMP ECHO_REPLY messages. The specific rule against the smurf attack would check for a spoofed source address associated with a broadcast destination address.

2.2 Uncompleted Connections

For a detailed analysis of the TCP/IP SYN flood attack, we refer the interested reader to previous works appeared in the literature (e.g., [11]). Critical factors for the success of this attack are the following:

1. the initiator of the bogus TCP/IP connections sends only one, i.e., the SYN, of the two messages, i.e., SYN and ACK, that it must send in order to complete the three-way handshake protocol. The initiator never replies with the expected ACK to the victim's SYN+ACK reply.

2. New bogus connections must be initiated by the attacking machine at a faster rate than the target machine's TCP timeout.

Because connection requests usually have spoofed source addresses of hosts that are not reachable from

[1] A statistical approach of the observed source addresses can be adopted to defeat possible changes of the computer IP address that aim at hiding the forged network traffic with spoofed source address.

the victim, the traffic originated by a SYN flood attempt could be blocked by the simple filter against spoofing the source addresses. However, since this is a mere implementation technicality that is usually performed in order to disguise the attacker's real identity, someone might try a SYN flood using the authentic source address. In this case, the attack should be blocked based on the conditions that characterize it.

In order to detect an excessive number of half-open TCP/IP connections, the middleware monitors all the TCP connections requests sent to each machine and keeps a counter, on a per user basis or on a system basis. If the number of half-open TCP/IP connections to a single machine and the rate at which they are initiated exceed given thresholds, the middleware completes all the pending requests with an RST packet and blocks further connections to that destination for a period of time. The duration of such a period can be computed to be larger or equal to the number of half-open connections times the largest timeout defined in the TCP/IP specifications. Because in case of legitimate connections the ACK packet would be sent timely, we believe that the chances to hurt regular users are minimum, although false positives are still possible.

2.3 Oversize Packets

Although the IP specifications indicate a maximum packet size of 65535 bytes, dimension checks are not enforced either at the source or at the destination to prevent the destination to overflow its buffer when it reassembles fragmented packets. Teardrop and Ping of Death are examples of attacks exploiting the oversize packet vulnerability. It is very easy for the filter to check the packet size and block packets that exceed the maximum packet size.

3 System Implementation

A prototype that implements the approach described in the previous section is currently under development at the Security and Network Laboratory of the Computer Science Department of the University of Milan [1]. The prototype is implemented on a Linux based system, kernel version 2.2.14 with firewalling extension.

It operates at the IP layer and is comprised of a static kernel module that applies packet filtering rules to the outgoing packets when they are ready to be passed on to the data link layer. The module is added to the native IP stack as a routine that manipulates a packet in the output chain, before the device driver receives it. This way the module monitors the outgoing traffic, without modifying the existing system. Because the module is compiled as a kernel static patch in order to prevent its easy removal, the kernel must be recompiled and the system rebooted if the set of rules identifying attacks changes. In order to take advantage of the in-depth security architecture provided by the firewalling extension, we register our module in the system firewall stack that control the output chains. Hostile packets are blocked by our filter. However, if a packet is "acceptable" according to the rules of our filter, it may still be blocked by the system firewall if any rule is implemented in it.

A data link layer module that could block hostile Ethernet frames could be developed in a similar way. Such a module would be useful to prevent attacks such as ARP cache poisoning that require data link layer malicious packet generation in order to bypass the legitimate ARP process [12]. We concentrate on the IP layer because attacks at this level are more popular. However, we successfully verified the applicability of our approach to the lower level so that a data link layer module is the next step in this project.

The packet flow is checked against attack signatures of known attacks and blocked when an attack attempt is detected, similarly to what an IDS does on the incoming traffic. The filter module operates on attack characterizations, i.e., the behavioral patterns typical of the various attacks. The more unique the attack pattern behavior, the more precise the action of the module, i.e., the less false negative and false positive signals the module will send. In case a hostile packet is detected, the default action is to drop it. Additional actions could be considered, such as logging all the intercepted traffic or letting the packet out anyway but signaling the superuser for further actions to be taken. Such a signaling part is handled by a separate module, e.g., by raising alarms, suspending the allegedly offending program, or logging the detected hostile activity.

4 Discussion and Conclusions

In this paper we have proposed a new strategy to deal with computer security problems based on limiting attacking capabilities. Its applicability to the specific cases of some network DoS attacks has been illustrated. The implementation of a prototype kernel level network filter for the various attacks considered is currently under development at our laboratory. Preliminary results on its effectiveness and impact on performance will be available in the future. Ongoing efforts are being put forth for the analysis and characterization of other types of attacks that can be dealt with following the disarming filters approach. The proposed approach raises various issues. The possibility of false positives may induce the system to drop packets of legitimate traffic. This could be a problem if the proposed filters were to be implemented on routers and other ISP network components in case of QoS oriented contracts. The issues of how to update the set of recognized attacks so as to incorporate new ones or rules that allow the filter to automatically detect new attacks should also be addressed. The use of encryption, such as with the IPSEC, SSL, and S/MIME protocols, is also worth investigating as it may affect the proposed approach. Furthermore, legal aspects of limiting attacking capacities need more detailed analysis.

References

[1] Bruschi D., Cavallaro L., Rosti E., "Less harm, less worry," submitted for publication, May 2000.

[2] CERT-CC, "TCP SYN flooding attacks and IP Spoofing attacks," CERT Advisory CA-96.21, http://www.cert.org, 1996-98.

[3] CERT-CC, "Denial of service attack via ping," CERT Advisory CA-96.26, http://www.cert.org, 1996-97.

[4] CERT-CC, "IP Denial of service attacks," CERT Advisory CA-97.28, http://www.cert.org, 1997-98.

[5] Ferguson P., Senie D., "Network ingress filtering: defeating denial of service attacks which employ IP source address spoofing," Network Working Group RFC 2267, http://www.rfc-editor.org/rfc/rfc2267.txt, January 1998.

[6] Garfinkel S., Spafford E., **Practical UNIX and Internet Security**, O'Reilly, 1996.

[7] Huegen C., "The latest in denial of service attacks: smurfing. Description and information to minimize effects," http://users.quadrunner.com/chuegen/smurf.cgi, last update Feb. 2000.

[8] Jones R., Kelly P., "Bounds checking for C," http://www-ala.doc.ic.ac.uk/phjk/BoundsChecking.html, July 1995.

[9] Kleinwaechter J., "The limitations of intrusion detection systems on high speed networks," presented at the "*First International Workshop on Recent Advances in Intrusion Detection (RAID)*," http://www.zurich.ibm.com/~dac/RAID98, Louvain La Neuve, Belgium, Sept. 1998.

[10] Neuman B.C., Ts'o T., "Kerberos: an authentication service for computer networks," *IEEE Communications*, Vol 32, No 9, pp 33-38, 1994.

[11] Schuba C.L., Krsul I.V., Kuhn M.G., Spafford E.H., Sundaram A., Zamboni D., "Analysis of a denial of service attack on TCP," *Proc. of the 1997 IEEE Symposium on Security and Privacy*, pp 208-223, Oakland, May 1997.

[12] Tripunitara M.V., Dutta P., "A middleware approach to asynchronous and backward compatible detection and prevention of ARP cache poisoning," *Proc. of the 15th Annual Computer Security Applications Conference*, pp 303-309, Dec. 1999.