Immunizing Servers from Buffer-Overflow Attacks

Zhenkai Liang, R. Sekar and Daniel C. DuVarney
Department of Computer Science
Stony Brook University
{zliang, sekar, dand}@cs.sunysb.edu

Buffer overflows have become the most common target for network-based attacks. Over 75% of the advisories issued by CERT Coordination Center last year were related to this class of attacks. Of particular significance is the choice of buffer overflows as the primary propagation mechanism used by worms. Due to the large number of insecure computers on the Internet, a buffer overflow vulnerability often results in an *automated attack* to the vulnerable system. A key characteristic of such automated attacks is that they are repetitive, i.e., multiple instances of the same attack may be launched many times against the same victim in quick succession.

Current technology for defending against buffer overflow attacks uses some form of guarding (as in StackGuard) or randomization (as in Address Obfuscation). Although these techniques can detect attacks before vital system resources, such as files, are compromised, they cannot protect the victim process itself, whose integrity is compromised prior to the time of detection. For this reason, the safest approach for recovery is to terminate the victim process. With repetitive attacks, such an approach will cause repeated server restarts, effectively rendering its service unavailable during periods of attack.

The process of defending against intrusions in computer systems is similar to that in biological systems, which is carried out by their immune system. The biological immune system, however, does a much better job in terms of efficiency. A biological immune system consists of two parts: the *innate immune system* and the *acquired immune system*. the innate immune system provides broad, non-specific protection against a whole range of pathogens, while the acquired immune system provides a specific and efficient protection against a particular type of pathogens. When a new pathogen invades a biological host, the acquired immune system does not have knowledge of the pathogen, and the biological host uses the innate immune system to respond to the invasion. The response relies on mechanisms such as *inflammatory response* that are not optimal in terms of targeting the damage at the pathogens. At the same time, the acquired immune system learns the characteristics of the new pathogen and synthesizes a response that specifically targets the pathogen without harm to the host itself. Once generated, the targeted response is present in the host in large concentration until the pathogen is wiped out, at which point, the concentration gradually falls down to a very low level. If the same pathogen is encountered in the future, the acquired immune system can respond very quickly to regenerate the same targeted response aggressively to wipe out the pathogen effectively. In a sense, innate immunity simply "buys time" for the acquired immune system, keeping the pathogen in check until a targeted response can be synthesized.

As we can see, the efficiency of biological immune system is achieved by the ability to learn new pathogens and synthesize targeted responses to them. That's a part missing in current approaches to defend against buffer overflow attacks. Our approach, called ARBOR (Adaptive Response to Buffer OveRflows), addresses this problem. It generates a rapid and automatic response to new buffer overflow attacks, and is a fully decentralized and lightweight approach which protects one or more processes. Based on the observation that attacks on networked servers arrive via input requests, ARBOR intercept input requests to a protected process, and screens out inputs deemed to contain attacks. In this way, ARBOR can stop an attack before it compromises the victim's integrity.

In our approach, we rely on a general-purpose intrusion detection system (IDS), coupled with coarse response such as a process restart, to serve as the "innate immune system." ARBOR, which corresponds to the acquired immune system, does not provide any protection initially against new attacks. When a new attack is encountered for the first time, the IDS response with a non-specific operation, usually terminating

the attacked process. At the same time, ARBOR immune response synthesis is initiated. In this phase, inputs associated with the attack are compared with benign inputs encountered previously. Based on this analysis, a filter is generated that can screen out attack-bearing input without (significantly) impacting benign inputs. The generated filter is then deployed in ARBOR. Once a filter is deployed, ARBOR keeps evaluating its effectiveness, which is determined in terms of the filter's ability to screen out attacks without impacting normal system activities. When the attack stops, or the filter becomes ineffective due to other reasons, the filter is disabled. The disabled filters are retained in "immune memory," so that when the same attack is encountered in the future, the response can be launched instantly.

By specializing the response to a particular attack, we increase the likelihood of finding a filter that can selectively block the attack without denying legitimate service requests. Moreover, continued evaluation of the filter protects against the possibility that the filter is evaded by attacks that change over time, as well as the danger of screening out future benign requests that happen to be different from those encountered in the past. The removal of filters at the disappearance of attacks provides assurance that service availability is restored to pre-attack state. Thus, even if the response mechanisms were less than optimal, and ended up discarding some legitimate requests, this will no longer happen when the attack stops by itself, or is stopped by other means such as upstream filtering on a firewall or network router. More generally, these safety features are designed to minimize the risk of "autoimmune disease," i.e., a situation where an over-zealous response contributes to a self-inflicted denial-of-service attack.

We have carried out comprehensive experimental evaluation of the approach, which establishes the following benefits of ARBOR.

- *Effectiveness against attacks prevalent in the "wild."* We collected all of the remote buffer overflow attacks published (together with exploit code) by SecurityFocus working on RedHat Linux (our experimental platform) during the period from 2001 to 2003. There were 8 such attacks, and ARBOR was effective in *immunizing against all of these attacks.*

- *Protection from repetitive attacks.* Our experiments show that ARBOR improves the availability of key servers (such as `httpd`, `ntpd` and `named`) by at least an order of magnitude, when exposed to repetitive attacks.

- *Insensitive to false positives.* For the above eight attacks, ARBOR does not produce any false alarms. Moreover, even when false alarms are artificially injected into ARBOR, their effects are minimal. In particular, it does not introduce catastrophic errors such as corruption of files.

- *Low overheads under normal operation.* Our approach introduces low overheads under normal operating conditions (under 5%), and modest overheads when under attack (under 15%).

- *Applicable to COTS software, without access to source.* ARBOR does not require any modifications to protected server, or even access to its source code.

Although our current approach is concerned only with buffer overflow attacks, we believe that it is more broadly applicable, such as to application-level denial-of-service attacks.