ARGUS SYSTEMS GROUP

PitBull LX

Is It Really Secured?

A Vulnerability Analysis of Operating System Hardening

White Paper
May 2002

Table of Contents

Introduction: Is It Really Secure?

The Solaris Operating Environment (and many other commercially available operating system) installs an insecure and exploitable system configuration by default. Many system administrators choose to "harden" their systems by removing unused or unwanted services and stripping down features. Built in features, as well as third party tools, can be used to harden the operating environment to minimize the risk of exploitation of the system.

Unfortunately, performing these hardening activities often gives users and administrators a false sense of security, and offers no concrete protection against many classes of attacks and areas of vulnerability. The goal of hardening a system is simply to make attacks on the system more difficult, but not impossible. Hardening does not stop attacks from taking place, nor can it prevent a user or attacking program from performing a successful exploit. Hardening a system does not attempt to add any security features or functionality, or attempt to strengthen security measures already in place.

OS Hardening

Techniques for hardening an operating system generally include the following steps.

- **Post installation package removal.** This step ensures only that the necessary binaries and files are on the system.
- **Elimination of Services.** Administrators may disable those services not required for operation of the machine.
- **File System Lockdown.** The file system can be locked down through the removal of SUID/SGID bits, changing some permission bits disallow specific access, and setting sticky bits on directories.
- **Configuration of Administrative Applications and Logins.** These critical accounts should be configured to only allow access for authorized users from specified locations.
- **Enable Passive Intrusion Detection/Auditing.** Passive event logging allows insight into and review of system activity.

Overview of Vulnerabilities Still Present

While performing each of these "lockdowns" offers some level of protection as part of a sound security model, they do not comprise a complete security model. A security analysis of a hardened OS would reveal additional areas of concern, outlined below.

- **Bugs are inevitable.** Even if notoriously "vulnerable" services are stripped out, attackers or rogue programs may discover new vulnerabilities in

established services or protocols. Consider the serious BIND or SMTP vulnerabilities, whose discoveries sent thousands of system administrators scrambling to patch systems long considered to be safe, well documented, and thoroughly tested.

- **Applications have power.** Applications and services typically need to access many different files, including system files, to operate on behalf of multiple users, or to use restricted network resources. Therefore, applications tend to be granted high-level administrative access to the system, allowing them to read and change any file on the system. Applications usually aren't programmed to run wild with these privileges, but if successfully attacked they can, and often do, take over the entire system.
- **Hijacked applications have power.** Applications can generate (spawn) other applications (child processes). The child process has the same level of access as the original process, so a child process can be just as destructive as its parent.
- **The root account can not be "stripped out".** The root account is the most widely abused account on the system, with the power to perform any action on the system. No amount of system hardening can remove the root account and its associated privileges. Root exploits can be conducted on any system, and can be used to perform any action.
- **Patches are ex post facto protection.** Even if all operating system patches are diligently and quickly applied to the systems, they remain vulnerable to the next-discovered (and yet unpatched) vulnerability. Vulnerabilities in standard operating systems are discovered on a continual basis, and the time from existence of a vulnerability to its discovery and the creation and dissemination of a patch can be weeks or even months. During this time, systems remain vulnerable to exploitation.

The most relevant and common attacks center around six issues: stack overflows, access to services, privilege and privileged accounts, networking resources, shared environments, and bugs in applications and services. None of these areas can be adequately addressed by performing system hardening. In some cases, hardening does not even perfunctorily address these vulnerabilities.

Addressing Additional System Vulnerabilities

Rather than removing system functionality or services to avoid vulnerability, PitBull products solve security dilemmas by implementing a new type of access control policy, managed and enforced by the controlling software of the computer – the operating system. PitBull's Domain-Based Access Control (DBAC) is a unique, patent-pending form of Mandatory Access Control that supports four types of access control: user, file, process, and network. PitBull's DBAC cannot

be overridden even if the user has been able to exploit services or programs to gain root-superuser privileges.

PitBull provides secure application technology to establish secure, isolated user environments and applies fine-grained administrative rights to compartments that cannot be penetrated by unauthorized users or by illegitimate means. PitBull's protections do not rely on known attack patterns, and apply regardless of whether bugs, flaws, and security holes exist in applications running on the system. PitBull security features cannot be found in standard operating systems and are not provided by any other security technology or combination of technologies. Because it operates at the operating system level, PitBull is able to prevent unauthorized access, even if the attacker (or attacking program) has gained 'root privilege' on the system.

Security functionality introduced by PitBull technology includes:

- Eliminating implicit power based on user identity (UID).
- Eliminating all-powerful root capabilities.
- Providing software subsystem isolation which can not be overridden or bypassed, using unique Domain Based Access Control (DBAC) technology.
- Extending the security layer beyond the perimeter and inside the DMZ to the "point of decision".
- Providing the tools and features needed while allowing standard applications to run in a highly secure environment without requiring their modification.

Protection Against Stack Overflows

Stack overflows are the main threat to most systems, and can either be exploitable remotely via the network, or locally. The most dangerous overflows are able to execute arbitrary code masquerading as the user for whom the application is running. It is also possible for attackers or rogue programs to modify data on the stack, causing an application to perform actions never intended, such as the modification of file names, user names, variables holding permission bit settings, IP addresses, port numbers, account names, execute file names, among others. These types of attacks can be devastating on a system, and cannot be deterred by hardening an operating system or by stripping out features and functionality.

The strength of PitBull LX security lies in its Secure Application Environment (SAE) technology. The SAE can be defined in a single word - containment. Processes, files, users, and network data can be confined into separate security compartments that can be completely isolated from other compartments. Through PitBull's SAE, applications are now 'contained' within subsystems. For example, on a PitBull LX system, if an attacker (or attacking program) exploits a stack or buffer overflow bug in a web application, he is not able to use that exploit to attack any other internal or external application, subsystem, or system.

Ability to Allow Full Access to Services

While hardening in conjunction with perimeter defenses attempt to address the security issue related to providing access to services, it is important to keep in mind that most attacks are based on abuse of allowed access to the system. There would be little point in deploying server machines without services running, and the ability for users to access and utilize those services. Therefore, stripping out services during system hardening is not a viable option for providing system protection. In fact, most systems provide multiple services; often services exist to provide access for system administration, for application administration, and to the application itself.

Instead, what is needed is a security technology like PitBull that provides an environment that protects the system no matter how weak the services or applications running on it. PitBull's Secure Application Environment applies controls that limit a service's access to files, networks, and other applications, in order to limit the damage from any exploit of an application, present or future. PitBull security mechanisms apply regardless of the level of administrative access a process has been granted.

Restriction of Privilege and Privileged Accounts

Even if a system is hardened, there exists a need to control privilege and privileged accounts on the system. Root has all privileges on a standard system; gaining control of a root process allows compromise of the entire machine. Even a non-privileged account is dangerous if an attacker can control a process, either through its environment or its faulty configuration, to perform unintended actions. For example, privileged accounts are often accessed by non-privileged users by breaking an suid/sgid binary.

To illustrate the damage potential in these types of cases, consider that many organizations have administrative personnel who each perform specific administrative functions. One administrator might be solely responsible for performing system backups, another might be responsible for installing patches, and yet another might be responsible for adding new users to the system. On a standard system, it would be necessary to give each administrator complete system access through the root/superuser account, the most powerful – and consequently most commonly abused – account on the system. Once an administrator has been granted root privileges on a standard system, they are able to perform any function including copying, altering, or deleting files; circumventing application level protections; or reconfiguring systems and applications.

With PitBull, administrative functions and otherwise privileged accounts can be separated so that no single user is able to perform every function on a system, even if some of those functions require root privilege. PitBull LX has the ability to restrict root by removing its privileges – subjecting the root account to security checks as if it were a standard user on the system. These security restrictions on

the root account are totally transparent to applications and provide 100% compatibility with standard system functions.

It is possible that system administrators might require full root privilege to perform some system function. This would be true in the case of upgrades to the system or in some configuration instances. PitBull allows an administrator to isolate a root account within a "virtual machine" such that only the subsystem being administered can be modified by that account; the rest of the system remains protected from exploitation. For example, these virtual machines can be used to allow administrative access to a system by various departments or groups, while restricting them to only the files and applications associated with their department.

In addition to administrative personnel, many applications are programmed to require privileged status to perform their proper functions. However, most applications which claim to require root privilege really only require a subset of the privileges available to root in order to function. PitBull offers a fine-grained privilege mechanism that allows restriction of the root account itself as well as the privileges any particular application needs, limiting the damage that can be caused by gaining control of a privileged application. PitBull's mechanisms are completely transparent, requiring no modifications to the applications themselves, and ensure that applications function no differently than they would have on a standard system.

Protection of Network Resources

OS hardening does not address the issue of networking resources. Any application has unrestricted access to the network (with the exception of binding to a privileged port). Applications need to be restricted in terms of what network resources they can access. For example, a system should be restricted in its ability to contact other machines on the network, to receive network traffic, to access backup systems, and other common activities. A need also exists to prevent an attacker from using a compromised application to attack other systems and applications either on the network or on the local machine. PitBull allows the precise definition of the resources an application may access by filtering network traffic based upon interfaces, IP addresses, ports, and protocols.

With its Network Domain functionality, Argus' PitBull LX fully satisfies any requirement to restrict access to network resources, or to restrict access based on where network traffic is destined for or from where it originates. A network access domain is a bit used to control access to a network object. The network access domain represents all types of access to a network object. Network access domains determine whether two processes can talk to each other through the local network or through other IPC mechanisms such as UNIX domain sockets. In order for two processes to communicate with one another they simply need to have one network access domain in common. Network access domains are also used on network interfaces to restrict a process's access to the network. PitBull's

networking functionality is administered through a firewall-style rule set. PitBull can be used to configure the network and can limit or completely prevent processes from accessing the network, even in the event of a process gaining superuser privilege.

PitBull's unique ability to extend network access control to the process level can be used to ensure that administrators can allow only specified access to applications or data, dependant on where a connection has originated or to which network a process is attempting access. PitBull security technologies allow administrators to implement further controls to limit messages coming from another network, particularly public networks. For instance, PitBull can be used to assure that network traffic coming from public networks can never be connected to internal databases. PitBull also implements controls so that computer systems accessible from other networks, cannot compromise the integrity of the organization's internal network.

Protection for Shared Environments

Because shared resources exist in the Unix environment (a good example being the /tmp directory), applications can sometimes be forced to write data into areas where others can read it. Attacks like sym link attacks, tmp file attacks, and core file harvesting rely on shared environments. Core dumps and other temporary files can contain sensitive information, or other information that could facilitate further attacks. An exploited application could be forced to export information to files locally or across the network. Because attacks such as these are application specific, without a thorough examination of each application's underlying code, there is no protection that can be offered through hardening a system

To ensure that all system resources are fully protected within shared environments, Mandatory Access Controls (MAC), which restricts access to only those resources for which an individual is authorized, are required. To fully protect shared environments, a solution must have the ability to provide fine-grained access controls based not only on user ID. These controls must also be applied to all processes spawned during a user's session.

However, DAC, the standard access control mechanism in commercial operating systems, offers no protection against an authorized user's ability to perform unauthorized activities. Standard Discretionary Access Controls (DAC) are insufficient to ensure that individual access can be restricted because DAC, by definition, relies on the user's voluntary compliance with the policies in place.

PitBull implements fine-grained MAC through the use of PitBull's File Access Domain functionality – a component of its Domain Based Access Controls. A file access domain includes a set of three bits used to control access to file system objects. The three bits of the file access domain represent read, write, and execute access. File access domains are placed on both file system objects and processes. In order for a given process to access a given file system object, the process must possess the exact access domain bit required for the type of access (read, write,

execute) for that specific file system object. This differs fundamentally from standard DAC in that DAC bases its access decisions solely on user identity, rather than on domain protections placed on files or resources.

File access domains are extremely useful for isolating files from certain processes or users. For example, few users on a system ever need to have write access to library files and other executables on the system. By placing a PitBull LX file access domain with write (w) restrictions on these files and never giving a user process this file access domain, it is possible to prevent a user from ever writing to these files. And since Domain Based Access Controls apply regardless of a user's intent and are unable to be circumvented by any user (including root), this protection applies even if the user changes users IDs to that of superuser through a root-compromise attack.

PitBull's four elements of Domain Based Access Controls (user, file, process, and network), give administrators powerful flexibility in protecting data and mission-critical applications. Administrators can grant proper, authorized access with a high degree of confidence that PitBull LX's DBAC will restrict users to only their authorized areas. All spawned processes inherit the same restrictions as the parent process, including instances where a process gains root privilege.

Protection Against Bugs in Applications and Services

Even in the best environments, bugs in applications still exist. A bug in an application can be as serious as an overflow exploit, and even the most secure application on a system can not protect against flaws exploited in a weaker application. For example, bugs may allow users to specify random files for uploading, random files for exec()'ing, or supply unexpected arguments to sub-programs and utilities.

The primary risk to most systems is the ability for an unauthorized process to gain control over resources an application is not intended to access. Through its Domain Based Access Controls, PitBull provides several mechanisms to reduce or eliminate this risk. PitBull's File Access Domains introduce a set of security attributes for the file system that are not dependent on the owner of the file. PitBull file access bits are associated with a process (which determine access to the file system) and persist across fork()'s, exec()'s, are not user dependent, and once enforced cannot be changed, even with root privilege. For complete protection, PitBull's Network Access Domains provide fine-grained control of network resources. PitBull associates security attributes with all network traffic on the system, allowing a security policy to be implemented by defining the exact networking resources that a particular subsystem can access. Networking access bits associated with processes are immutable, even with root privilege. Additionally, PitBull provides a mechanism that relies upon file system and networking controls to allow restricted logins (i.e. the shell or process created when a user logs into the system) protected by PitBull. As a result of restricted logins, user environments can be created to limit the access to a particular

subsystem regardless of whether a user or process is able to gain root privilege (e.g. through the su command.)

Summary

While OS hardening is a prudent step in securing a system, relying solely on stripping out features and services to provide a robust security environment is a risky proposition. Common vulnerabilities exist that cannot be addressed by hardening tools alone. PitBull provides the necessary mechanisms to address these vulnerabilities and provides the level of security demanded on a production system in any environment or architecture.