

PitBull LX

Secure Application Environment

White Paper
September 2004



PITBULL
LX

COPYRIGHT

©1997-2004 Innovative Security Systems, Inc. 1809 Woodfield Drive, Savoy, Illinois, 61874 U.S.A. All rights reserved. This product and related documentation are protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or related documentation may be reproduced in any form by any means without prior written authorization of Innovative Security Systems, Inc. and its licensors, if any.

DISCLAIMER

THIS PUBLICATION IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

NOTICE TO USER

This publication could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. These changes will be incorporated in new editions of the publication. Innovative Security Systems, Inc. may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

TRADEMARKS

PitBull is a registered trademark of Innovative Security Systems, Inc.

Innovative Security Systems, Argus Systems Group, and PitBull LX and their respective logos are trademarks of Innovative Security Systems, Inc.

All brand names and product names used in this document are trade names, service marks, trademarks, or registered trademarks of their respective owners.

Table of Contents

THE PROBLEM: APPLICATION SECURITY FLAWS.....	1
A BUG’S LIFE.....	1
WHAT’S THE ANSWER?.....	2
APPLICATION SECURITY.....	2
PITBULL LX DESIGN.....	3
DOMAIN-BASED ACCESS CONTROL.....	4
FILE DOMAINS	4
NETWORK DOMAINS.....	5
SECURITY FLAGS	6
PUTTING IT ALL TOGETHER.....	6
OS-LEVEL PROTECTION.....	7
SUMMARY.....	7
FOR MORE INFORMATION.....	8

The Problem: Application Security Flaws

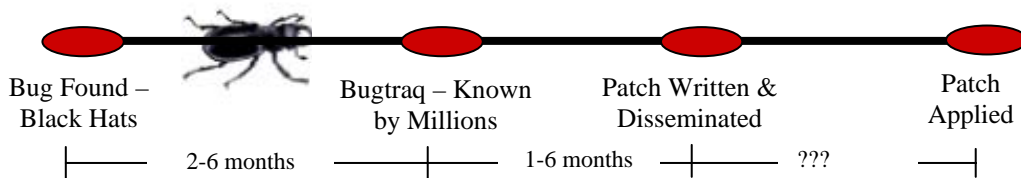
In today’s e-business environment there is an increasing reliance upon third-party application software, for many business reasons. Commercial, off-the-shelf applications improve time-to-market, lower development costs, and help companies integrate more smoothly with other organizations and environments. But once an e-business enters the production phase and its servers are connected to the outside world via the Internet, application software introduces serious security vulnerabilities that can immediately cripple the e-business.

Why? It is a well-known fact that software programs of all types contain security flaws (bugs). Every day we read about bugs that are being discovered that, when successfully exploited, can give attackers root access (full administrative powers) to mission critical servers. A bug in a web server application, for example, which is open to receiving traffic from virtually anyone and everyone, provides a malicious user with a launching pad to attack back-end systems and even your most critical data. To make matters worse, it no longer requires a great deal of expertise to exploit these vulnerabilities. Sophisticated hacking tools that exploit application bugs are readily available to even novice hackers (script kiddies), further expanding this threat.

Unfortunately this problem is not going away anytime soon. In fact, due to increasing time-to-market pressures, over-worked security staffs and over-stretched IT budgets, there will be a growing dependency upon flawed software applications, adding to the number of security holes into today’s e-businesses.

But the problem is not associated with only commercial applications. According to Jody Patilla (a consultant for Metases) in a recent eWeek column, “...in my opinion the more serious threat comes from security vulnerabilities in badly written home-grown web applications—Common Gateway Interface scripts, server-side executables, active scripting and so forth. All too often, authentication methods are weak, session information is exposed, user input isn't properly validated—the possibilities are endless. These are the holes that are most likely to expose customer or business proprietary data. What's more, they are harder to find and fix because there's no vendor to issue a patch, and, unfortunately, they are plentiful. Without adequate training, in-house or contract development teams may not even be aware of problems.”

A Bug’s Life



Pick any popular web application, and it is bound to have multiple security flaws. These flaws often exist for long periods of time before they are ever made public. Since they represent a hacker’s most common means of breaching a system or network, it is often the hacker (Black Hat) community that first discovers these vulnerabilities. After hackers have had their fun (at someone else’s expense) the bug will often get posted on one of the popular bug-tracking sites, such as Bugtraq. Eventually the software manufacturer will

release a patch to plug the hole and will notify its customers. The patch may or may not be applied by the end-user. The lesson to learn is that even if a company is diligent about applying patches and plugging security holes, it is still vulnerable—in some cases for months or even years before a bug is made public and a patch can be applied.

What's the Answer?

Security experts will often say that the solution to this problem is to stay constantly on the alert, monitoring your applications for security holes and applying patches and bug fixes when they immediately become available. Although this is good advice, for many companies this is a daunting task. Hundreds of servers running a myriad of applications can make this difficult, if not impossible to fully carry out. Many IT and information security departments are already understaffed and overworked. Staying 100% up-to-date on all patches is not their highest priority. And even if it was, the magnitude of the task alone makes it impractical. Plus, no amount of diligence will protect an organization against bugs that have not yet been made public, or where a fix has not yet been released. This is similar to a disease, where people first have to be infected before a vaccine can be developed. Hoping that someone else gets infected before you do is no way to run a business—especially when preventive security solutions are available today. Leading-edge security practitioners are already benefiting from these new technologies via lowered risks and reduced administrative efforts.

Another common response to application security flaws is to add more and more costly layers of perimeter security such as firewalls and intrusion detection systems. Although these technologies can provide valuable security functionality, they do not and cannot address the problem of application bug exploits. Firewalls may limit direct access to back-end networks or servers, but they do allow access to web applications. In fact, the minute a company connects a web server to the Internet at www.company.com, visitors to that website are by definition beyond the perimeter firewall. If bugs exist in those web applications, your sensitive internal systems and data are vulnerable to successful buffer overflow attacks, worms, and a myriad of other exploits.

Intrusion detection systems (IDS) are also limited due to their mode of operation. Most IDS look for known attack patterns, and when they recognize an attack, send out alarms so the attack can be thwarted. Often, by the time IDS responds to an attack, it is too late—the damage has already been done. Plus, IDS (even active IDS or host-based IDS) provides little to no protection against unknown attacks or for encrypted channels. New attack methods can penetrate even a properly-configured intrusion detection system.

Application Security

“Despite a plethora of sophisticated security solutions on the market today, very few of these products address the problem of application security... Once the external user is permitted access to the corporate network, he or she often has a free pass to the databases behind the application. This situation has created the need for an entirely new set of solutions especially geared toward securing what we call the ‘last mile’ of the e-commerce transaction, where a vast majority of security attacks take place.”

Bear Stearns, “Internet Security”, June 2001.

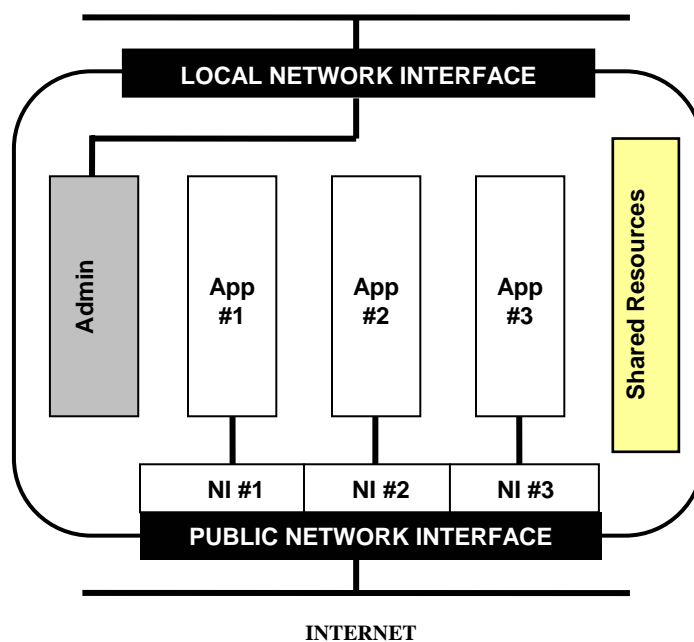
A new category of security solutions is emerging to address the problems created by application security flaws. This new category, which Bear Stearns simply calls

‘application security’, is projected to be a \$3.5 billion market by 2005. Argus Systems Group, a leader in Internet security solutions since 1993, has launched PitBull LX into the application security space and has become the leading solution in this category.

PitBull LX Design

The strength of PitBull LX lies in its Secure Application Environment (SAE) technology. The SAE can be defined in a single word: *containment*. Processes, files, users, and network interfaces can be confined into separate security compartments that can be completely isolated from other compartments. In PitBull LX terminology, these compartments are referred to as domains. Multiple domains can be created on a machine, converting a single server into a system supporting hundreds of *virtual machines*, each with its own Secure Application Environment supporting secure applications and secure services.

Through PitBull’s SAE, applications are now contained in these subsystems. On a PitBull LX system, if an attacker exploits a bug in a web application he is not able to use that exploit to attack any other internal or external application, subsystem, or system. It is as if the attacker is locked inside a jail cell (domain) with no way out, even if the bug exploit results in the attacker gaining root access. The root attacker could not override any of the LX domain attributes and could not escape the LX domain.



PitBull LX’s Secure Application Environment supports the secure hosting of multiple applications on a single machine. Not only does this provide robust security, it significantly lowers costs by improving server utilization and lowering administration overhead.

PitBull’s SAE is a powerful solution in dealing with application security flaws. No longer does a security professional have to stay up at night, hoping and praying that he has installed all the latest security patches. With PitBull LX installed, if a particular patch has

been overlooked, it no longer presents a life-threatening risk to an organization. All bug exploits, both known and unknown, regardless of their source or intent, will be securely contained by PitBull's Secure Application Environment.

Domain-Based Access Control

In order to create SAEs, PitBull LX utilizes a concept called Domain-Based Access Control (DBAC). DBAC controls access based on a new set of attributes, called *domains*, that can be applied to users, files, processes, and networking. This mechanism is both powerful and flexibility when it comes to protecting data and mission-critical applications. DBAC allows administrators to grant proper, authorized access to those that need it with a great degree of confidence, trusting PitBull LX to restrict users and processes to their authorized areas.

PitBull LX supports two types of domains: *file domains* and *network domains*. These domains can be placed on a process, and whenever the process attempts to access an object (such as a file or a network resource) the process's domains are compared to the domains on the resource to see if the access can occur. The system administrator can choose to have some users and processes run with PitBull LX restrictions (this is referred to as being *LX aware*) and to have some users and processes run completely unaffected by PitBull LX (referred to as being *LX unaware*).

File Domains

A file domain is represented by a symbolic name and an access mode set. The name is chosen by the system administrator and the access mode is in the form *rwX* (read, write, execute), the same as used in standard Unix for permission bits. A system can support 1,024 different file domains.

For example, a file could be given the domain *sys(-w-)*. This domain does not limit read access or execute access in any way, but in order to write (modify) this file, a process must have write access in the *sys* domain. So a process with a *sys(rw-)* or *sys(-w-)* or *sys(rwx)* domain would be able to modify the file, but a process with no *sys* domain or only *sys(r-x)* would be unable to modify it. If an administrator were to put *sys(-w-)* on all system files, directories, and utilities and if the administrator were to not give any process the *sys* domain, then the system resources would be absolutely protected from any LX aware process, even if that process was running as superuser or root.

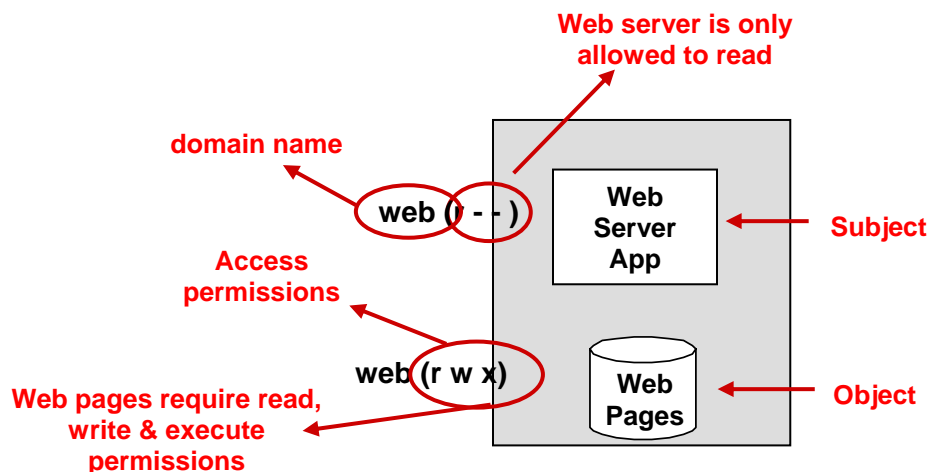
If a file has more than one domain, a process must have all of those domains (with the proper access mode) in order to be able to access the file. So in general, the more domains on a file, the more protected it is; and the more domains on a process, the more access rights it has.

The following are typical file domains found on many PitBull LX systems.

- *sys(--x)* used to prevent access to sensitive programs and utilities
- *sys(-w-)* used to maintain integrity of critical files
- *web(r-x)* used to isolate web site files from all other files

Protecting internet-based servers is a typical use of PitBull LX. For example, if a web server is to read its web page files, the web server must have a domain name of *web(r--)*, compared to the web pages with its domain name of *web(rwx)*. In this simple scenario, the web server is only allowed to read the web pages. Even if the web server is

compromised it will not be able to expand its access permissions, but will always be locked in a read-only mode. This step alone protects web pages from defacement.



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 restrictions on these files and never giving a user process this file domain, it is possible to prevent a user from ever writing to these files. This holds true even if the user changes user IDs to that of superuser by executing a buffer overflow attack on a setuid root program.

DBAC prohibits users from accessing off-limits domains by assigning them file domains upon login. This is accomplished via Pluggable Authentication Modules. Based upon their corporate security policy, administrators can easily assign multiple file domains to individual user accounts. If a user does not possess a particular file domain, LX will block them from gaining access to any files or processes within that domain. File access domains can also be assigned to groups of users. For example, any user coming in from the Internet can automatically be assigned a single domain of *inet*. As long as back-end applications do not have this domain name, Internet users will be barred from accessing sensitive internal data. DBAC does not allow the user to shed this *inet* domain, regardless of his activities, even malicious ones.

Network Domains

A network domain is used to control access to a network object. It is represented as *domain_name*(net), where *domain_name* is the symbolic name for the domain and (net) signifies that it applies to network objects. Network domains apply to both network objects and processes. There are 512 unique user-defined network domains available. Examples include:

- eth0(net) – access to eth0 domain
- port80(net) – access to port80 domain

Network domains determine whether two processes can communicate with each other. In order for this to happen they need to have *only one* network access domain in common. Network access domains are also used on network interfaces to restrict a process' access

to the network. Firewall-style rules—called *NetRules*—are 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 status.

NetRules work in tandem with network domains, providing an even finer-grained method of network access. NetRules can be assigned on any combination of the following: destination address, destination port, source address, source port, interface and protocol. For example, let's consider a web server that has a network domain of *web(net)*, that should only be able to accept traffic from port 80. Under PitBull LX we can assign a NetRule that will allow all port 80 traffic to reach the web server by automatically assigning port 80 the *web(net)* domain, while rejecting all traffic from any other port by assigning all other ports a different network domain, such as *Netdenied(net)*. Any traffic now coming in from port 2000, for example, will be assigned the *Netdenied(net)* domain and will be automatically dropped.

Security Flags

File security flags are special flags that can be placed on files to dictate certain behaviors. These behaviors can tell PitBull LX whether it is protecting a file, whether PitBull LX security should automatically be placed on files that are created in a particular directory, as well as cause special security actions to be taken when a file is executed. Many different security flags are available. They are designed to provide a flexible way of applying system-wide security with minimal effort. Security flags can also be associated with processes and networking objects. Security flags are one of the key tools used to restrict superuser privileges.

One example of this feature is the security flag `ASG_SEC_ACC_FS`. This flag restricts access to files that are not protected by PitBull LX. If this flag was applied to a web server, the web server would not be allowed to access any file system object on the server that was not LX-aware. The benefit of this flag is that through one simple step hundreds of files can be isolated from an application or process without having to make any changes or modifications to the files themselves.

Putting it All Together: Securing an Apache Web Server

With PitBull LX, locking down an Apache web server can be accomplished with four simple steps. For more information on this procedure, consult the PitBull LX Quickstart Demonstration Guide.

1. **Create file domains.** Assign the same file domain (*web*) to both the web server and the web files. The web server should be given only read access (*web(r-)*) so that it cannot write to or execute the web pages. This protects the web pages from defacement. The web page files should be given read, write, and execute access permissions (*web(rwx)*) so they can restrict these types of access when necessary.
2. **Run Argus lockdown script for Apache.** An Apache lockdown script is delivered with the PitBull LX software. This script basically does two things: 1) assigns the web server a network domain of *web(net)*, and 2) applies a security flag to the web server that will not allow Apache to access any file system object that is not protected by LX. This flag protects every file on the system without having to apply LX security to each and every file. Now the web server can only do one thing: read its web pages.

3. **Apply NetRules.** One NetRule needs to be applied, which allows the web server to only communicate on port 80. Traffic from any other port will be automatically assigned a *netdenied(net)* network domain. This single NetRule blocks all communication except that which comes in on port 80.
4. **Assign an Administrator.** By placing an entry in the LX user database, an administrator can be defined for the web domain by assigning him the file domain of *web(rwx)*.

PitBull LX is a straightforward solution that is easy to install and deploy. In addition to web servers, it is an ideal solution for securing application servers, DNS servers, mail servers, database servers, ftp servers, security servers (firewalls, PKI, IDS), and e-commerce servers. Lockdown scripts are available for a variety of popular applications, which streamline the configuration process. For more information about applying LX technology to your specific architecture, contact Argus at info@argus-sysetms.com.

OS-Level Protection

PitBull LX is a kernel-loadable module available for Solaris, AIX, and multiple distributions of Linux. Unlike most security technologies that reside in application space, PitBull resides at the OS-layer. It is a well-known but often overlooked fact that if an attacker takes over the operating system, all security mechanisms in application space can be bypassed and rendered useless. By locking down the OS, PitBull's Secure Application Environment is based upon a very solid foundation. The Domain-Based Access Controls are enforced by the OS, which keeps them from being circumvented even in the case of someone gaining root access.

Summary

PitBull LX's Secure Application Environment is a powerful tool in the fight against application security flaws. It delivers a level of security that cannot be achieved through the use of traditional solutions. Through the deployment of PitBull LX, organizations can:

- Capture application bugs in security compartments
- Stop web-facing servers from becoming bridgeheads into internal systems
- Grant the right type of access to those that need it -- deny access to others
- Stop the propagation of worms & other rogue processes
- Restrict root (superuser) powers

PitBull LX was put to the test in eWeek's Open Hack III competition. Four servers (shell, DNS, web, and e-commerce) were connected to the Internet without any protection from firewalls or IDS. PitBull from Argus was the only security solution deployed. User login access was provided to the servers. After seventeen days and over 5 million hacking attempts, PitBull LX stood firm. Even after a successful root exploit, the hacker was unable to break out of PitBull's Secure Application Environment.

For more information

For more information about Argus Systems Group and PitBull:

Email: info@argus-systems.com

Website: www.argus-systems.com

Phone: +01 217-355-6308

Fax: +01 217-355-1433

Address: Argus Systems Group
1809 Woodfield Drive
Savoy, IL 61874 USA