PitBull .comPack

OS-level Security for Solaris and AIX

White Paper
March 2001

PITBULL .COMPACK ™

INTRUSION PREVENTION SYSTEM

Table of Contents

Why do you need Intrusion Prevention System security?

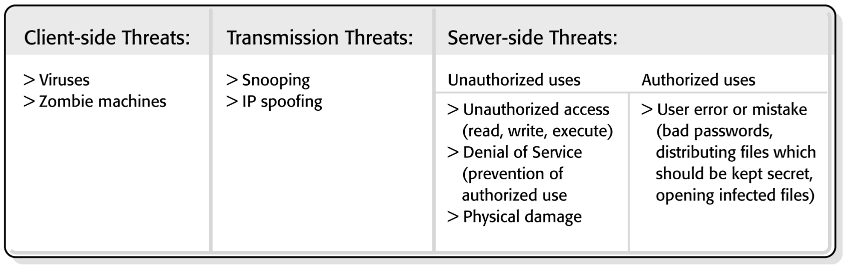
Ten years ago, the Internet business world was just beginning. The distinctions between where internal systems ended and the Internet began was reasonably clear. Today, that's no longer the case. When your business is your website, or your online database, or requires instant data exchange with a branch office which is physically located 2,000 miles away, the distinction between your business and the Internet blurs to the point where they become indistinguishable.

Firewalls, encryption, and intrusion detection software may help, but are not sufficient in themselves. When the firewall has to permit traffic through to access the web server, it opens a hole to potentially permit unauthorized access to your internal systems. If data is encrypted in transmission but is stored in an accessible form on your server, anyone who can perform a root exploit on your server doesn't need to bother with intercepting packets and breaking the encryption. The essential problem with intrusion detection is that the intrusion has already occurred by the time there is enough of a pattern for the software to identify. The same is true of anti-virus software: in order to produce the vaccine, someone must have been infected. At the speed with which modern computer viruses such as Melissa and ILoveYou propagate and distribute themselves, this response method isn't fast enough to guarantee that your business won't be one of the victims.

Your first line of defense needs to be as close as possible to the data and applications being protected. And it needs to be deeper than any user can access, even root. Argus's PitBull Intrusion Prevention System, based on trusted operating system technology, both secures your computer at the kernel level and improves the security of your other application-based security systems. By placing the security decisions within the kernel itself, no application can circumvent the security decision-making process. By separating root's power into many different, more limited privileges, and preventing the root user from acquiring more privileges, you no longer need to worry about root exploits in applications since root no longer has any real power. By compartmentalizing each application with sensitivity labels (which, unlike traditional UNIX security, are not dependent on the user ID and cannot be overridden by root), an exploit which gains control of one application can't be used as a springboard to control another. And with the ability to compartmentalize different elements of the same application, such as a web server, even exploits within an application can often be prevented from damaging the application itself.

Threats and countermeasures

No matter what species of attack takes place, the eventual target points remain the same. In an Internet security breach, there are three primary points of failure: the client's system, the packet in transmission, and your system.



Client-side Threats:	Transmission Threats:	Server-side Threats:	
		Unauthorized uses	Authorized uses
> Viruses > Zombie machines	> Snooping > IP spoofing	> Unauthorized access (read, write, execute) > Denial of Service (prevention of authorized use > Physical damage	> User error or mistake (bad passwords, distributing files which should be kept secret, opening infected files)

FIGURE 1. **Three Catergories of Threat**

The client's system

Unfortunately, in a Web-based business model, there is very little a business can do to guarantee that the computers which are being used to connect to their website and/or databases have not been compromised. The rapid and global transmission of recent viruses such as Melissa and ILoveYou testify that anti-virus software has the same limitations as regular human vaccine: someone has to suffer the infection before the doctors can create a cure. And the use of "zombie" machines to perform coordinated denial of service attacks reinforce the problem: you can't count on all incoming requests being friendly, but you also can't block all requests if you want to perform Internet-based business. Since it's impossible to guarantee that the other system has not been compromised, the best thing you can do for your own security is to ensure that your operating system and file systems are as bullet-proof as possible.

If the ILoveYou virus had been written for a UNIX platform, Argus's PitBull security system could have drastically reduced the damage performed by the virus. In a properly compartmentalized system, the virus might still have had the ability to redistribute itself, but it would have been prohibited from corrupting other files on the system. The reason is that in a PitBull system, processes are allowed to read files stored at lower sensitivity labels, but can only write to files stored at the same sensitivity label. Therefore, although the retransmission part of the virus might still have been permitted (because it would be received at the level of the mail system), the damage it tried to cause to other parts of the system would be blocked (because other applications and files would be stored at different sensitivity labels). The security principles behind this will be explained later in this paper.

In transit

Another common target is the transmission channel. Since data packets have to travel through several computers to be routed to the correct destination, packet interception and reading is not much of a challenge unless the packet is encrypted. This is why sensitive information should be encrypted over any network channel. Another common attack is for a machine to pretend it has another machine's IP address ("spoofing"), in order to either send or receive information which should have come from the other machine. Strong authentication methods and signed keys provided by trusted vendors are used to combat this problem, but an additional concern is the integrity of the trusted vendors' databases and authentication sites.

PitBull Foundation™ ships with Advanced Secure Networking, which provides a means to insert security information into standard networking traffic -- and require corresponding security information in packets it receives, if the system security policy requires it. PitBull .comPack™ offers Trusted SSH, a sensitivity-level-aware version of F-Secure's SSH, which provides encrypted telnet and ftp-like traffic. PitBull .comPack also offers IPsec, which covers three functional areas of security: authentication, confidentiality, and key management. Finally, PitBull Foundation's Pluggable Authentication Module (PAM) support permits integration of a wide range of third-party authentication solutions, including smart cards and biometrics. PitBull systems both provide native network security mechanisms and enhance the security of any additional mechanisms your business requires.

Your system

The widest variety of attacks available are aimed at the system providing the service rather than the connection or client computers. And not all system compromises are due to attack; others are due to simple mistakes. In any case, the types of system compromises available fall into four primary categories:

- Unauthorized read, write, or execute access
- Prevention of authorized access (denial of service)
- Authorized users making either mistaken or malicious decisions
- Physical damage to hardware

Unauthorized read, write, or execute access

Most exploits in existence take the unauthorized access approach. Although the methods vary widely -- buffer overflows, CGI exploits, root exploits, etc. -- the hacker's desired end result is access. The application-level solutions are as varied as the attacks: firewalls, intrusion detection software, access control lists, and so on.

While Argus does *not* claim that all application-level security is useless, we *do* assert that almost any amount of application-level security is insufficient by itself. The reason is that a root exploit can be used to control any standard UNIX system -- including its security applications. A root exploit cannot be used to control a PitBull system. The PitBull system provides operating-system-level security and modifies the kernel so that UID 0 no longer has the innate powers associated with that identity on standard UNIX. However, almost all commercial off-the-shelf software will run on a PitBull-secured system without modification, because the PitBull system provides integrators with the ability to assign specific limited privileges to executable files, and to control their use.

Prevention of authorized access (denial of service)

Another common attack is the denial of service attack, one variety of which made national headlines in February 2000 when several major Internet businesses were disabled for several days. This kind of attack is less directly useful for a hacker, because it does not yield any information, but it is effective in terms of sheer nuisance value. The distributed denial of service attack types used in the February 2000 events require a hacker to have compromised several systems before launching the assault. So far, no single solution to this problem has been found. Countermeasures have consisted of pattern-matching on either the network level or the application level in order to apply filters or bandwidth limitations.

For Internet-based sales sites and other applications which are intended for worldwide access, where the accepted IP addresses cannot be limited by default, PitBull will not prevent you from being targeted by denial of service attacks. However, PitBull will prevent your site from being used as a launchpad for a denial of service attack; the launchpad sites are often liable for prosecution. In addition, for Internet-based intra-company applications such as branch office communications and worldwide secure database access, PitBull provides several countermeasures to denial of service attacks. These countermeasures include Virtual Private Networking and Advanced Secure Networking's packet filtering options. In a situation where you wish to restrict traffic to known hosts, and to verify the identities of the connecting machines, PitBull permits effortless rejection of packets from unapproved sources.

Authorized users making incorrect decisions

One of the most difficult problems to solve is that of authorized users making mistakes. Examples of this include writing down or telling another user a password, opening infected files from known or unknown sources, making a file world-readable or world-writable for convenience, opening a door through a firewall for home-based access, and so on. An even more difficult problem is a knowledgeable user with a grudge, such as a recently-fired system operator or administrator. In the standard UNIX security system, the dependence on user integrity is a substantial weakness. Under the assumption that no one would give file access to an unauthorized user, access controls are left to the user's discretion. Under the assumption that only authorized users would be able to become root, root is all-powerful. Neither of these assumptions hold up in the era of modern cybercrimes.

Although no software system can prevent a user from writing down his password and handing it to another user, PitBull systems come closer than most to solving the problem of incorrect user security decisions. Compartmentalization can be used to restrict individual users to a small area of the system, so that a compromise of one user's account can't be used to damage others. The Foureyes program can require confirmation of actions by designated users, so that certain tasks can only be performed after a second user has agreed to it. And PitBull's inherent division of power splits system administration tasks and abilities among multiple users (security system administration, system administration, and system operations).

In addition, PitBull builds on each operating system's native auditing capabilities to permit tracking of both standard auditable events and security-specific ones. Additional audit classes and events are available for site-specific use.

Physical damage

No software in the world is able to prevent all types of physical damage to hardware, but many are designed to simplify recovery in the event of a hard disk failure or other drastic hardware loss. In addition to native backup and restore utilities, PitBull provides a simple way to include the system security information with any third-party backup and restore software: the integrity database system, which is used both to check for system alterations during ordinary operation and to record the correct security settings for designated system files and devices to assist with the backup and restore process. The components of PitBull Foundation and PitBull .comPack install with default integrity databases in place; the site administrator can create integrity databases for any additional software whose security information should be recorded and regularly reviewed.

Trusted OS Security: principles and practice

There are four primary principles behind trusted operating system security:

- **Information compartmentalization**. Restrict access to information in a way that is not dependent on the user ID for security control. Provide a means of preventing any user on the system, including root, from viewing or modifying information which the user is not cleared to have. Prevent compromises in one application from being used as a springboard into another unrelated application.
- **Role compartmentalization.** No one user on the system should be able to perform all system events. Access to a root shell should not give control of the entire system. Confirmation from two users is strongly recommended for important system actions such as addition of users, addition of devices, and system reboots, to limit the amount of damage which can be done by a compromise of any individual user account.
- **Least privilege.** Processes should only be permitted to perform their designated tasks; for example, a mail system process (even one running as root) should not be permitted to modify a web server's files. Processes should not have any privileges which are not essential to their operation; for example, a web server (even one running as root) should not be permitted to modify any files other than the web server's own access and error logs.
- **Kernel-level enforcement.** Security should be performed at a level which cannot be bypassed by any user-level actions, and at a level as close to the application which is being protected as possible. Kernel-level security enforcement ensures that access decisions are made at a level that users cannot circumvent, and these access decisions directly precede the application's actions.

PitBull Foundation: How we apply those principles

PitBull Foundation, the trusted operating system component of Argus' Intrusion Prevention System software, addresses each of these requirements for trusted operating system security:

Information Compartmentalization

Mandatory Access Control (MAC) is PitBull's answer to the information compartmentalization requirement. Unlike standard UNIX's Discretionary Access Control (DAC, more familiar as permission bits and access control lists), MAC is not dependent on the discretion of the user. A user can own a file which has read, write, and execute permissions available to everyone, but under MAC, if the user is not cleared for the information in the file, the user can't touch it. The key is **sensitivity labels** (**SLs**). Every object on the system, including both files and processes, has a sensitivity label; some objects, such as directories, can have more than one SL in order to define an access range. And unlike standard DAC, MAC restrictions cannot be overruled by a root-owned process.

There are two components to a sensitivity label: classifications and compartments. Every sensitivity label must have one classification, which is the hierarchical component. For example, Top Secret is a higher classification than Secret, which is higher than Confidential. However, if the classification were the only component of a sensitivity label, there would be no way to prevent a Top Secret user from reading (though not writing) every file on the system; therefore, the second component of an SL, the compartments, are non-hierarchical. For example, compartment A is neither higher nor lower than compartment B. A sensitivity label can have no compartments or any number of compartments (up to 1023, the system limit).

The use of both classifications and compartments in SLs create three possible dominance relationships among files and processes: **dominant**, **equal**, and **disjoint** labels. A process can read but not write any file which it dominates but does not equal. For example, a process with a Top Secret SL can read but not write a file with a Confidential SL. Writes are only permitted when the process's SL equals the file's SL. Disjoint labels are used for pure compartmentalization, to prevent all access between the two areas. For example, a Top Secret A process can neither read nor write a Confidential A B file, because the process's SL does not have access to the B compartment.

Users are also assigned sensitivity levels; each user account has three sensitivity labels associated with it, which define the user's **clearance**. A clearance includes a minimum SL, a maximum SL, and a default SL. These SLs define the range at which the user can log in to the system; the default SL will be used unless another SL within the user's range is specified in a console or TSSH connection. (TSSH is described in more detail in the PitBull .comPack section of this paper.)

The level at which the user logs in will determine the level at which each of the user's shell's child processes will be created, and most users are not authorized to change their processes' SLs (even within their clearance) after logging in. Processes inherit their SLs from the parent process -- including root-owned processes, and, as mentioned above, root is subject to MAC controls in the same way that any ordinary system user is.

Another feature PitBull Foundation offers for information compartmentalization is partitioned directories. Ordinary directories can be single-level (minimum and maximum SLs set equal) or multiple-level (a range of SLs permitted between the minimum and maximum). Partitioned directories appear to be single-level to any single user, but behave like multiple-level directories. Within the real directory there are several virtual subdirectories, each of which operates at a single SL. A user whose shell is operating at Top Secret who enters the partitioned directory will arrive in the Top Secret virtual subdirectory, and will not be allowed to see any files at other labels in any of the other subdirectories even if his clearance dominates the lower labels. Similarly, a user whose shell is operating at Confidential who enters the partitioned directory will see only the contents of the Confidential virtual subdirectory.

In practical terms, this means that root exploits are no longer useful to hackers. Even if a web server is running as root, the process of the web server in a properly compartmentalized system will have a SL which does not equal that of any other application or group of files, including its own HTML and configuration files. A hacker who spawns a new shell based on a web server exploit can only write at the SL at which the web server ran.

Although the web server process's SL would dominate the SLs of its HTML and configuration files in order to be able to read them, the only files which the hacker could write to are the access and error logs. And the web server's process will not have the privileges required to raise or lower SLs.

Role compartmentalization

A combination of privileges and authorizations are used to enforce role compartmentalization. On a PitBull system, root's powers have been splintered into many smaller, more limited abilities called "privileges," which are no longer inherently associated with UID 0. In fact, by default the root user cannot use any privileges which are not already associated with a given process or executable, because of a kernel security flag which prevents root from using authorizations.

Authorizations are assigned to users; privileges belong to processes and executable files. Thanks to the combinations of authorized privileges/privileged authorizations and innate privileges/access authorizations, if a user is not authorized for a privilege, he will not be able to use that privilege even if he is permitted to execute a privileged file.

Root is not the only user whose powers have been restricted. In the default installation, system administration responsibilities are divided among users who are assigned the Information Systems Security Officer (ISSO) authorization, the System Administrator (SA) authorization, and the System Operator (SO) authorization. Broadly speaking, the ISSO controls PitBull security functions, the SA controls ordinary UNIX functions, and the SO controls devices and hardware. The addition of users or software requires cooperation among these three roles, so that a compromise of one powerful account will not necessarily compromise the entire system.

Least privilege

The principle of least privilege is closely associated with power compartmentalization. Least privilege means that a process or executable should only have the minimum necessary privileges and for only as long as those privileges are required. It also means that a user should have only the authorizations which are required for the performance of his duties, and should have the least possible clearance range compatible with those duties.

This principle extends into all aspects of the PitBull system design. Unlike SLs, a child process's privilege set is not directly equal to that of the parent; the child process's privilege set is calculated with various factors, including the executable's privileges and the user's authorizations. The different types of privilege sets available permit fine-grained control of both file access and privilege use.

Kernel-level enforcement

The purpose of kernel-level enforcement is both to reduce system overhead by bringing the security decisions as close as possible to the resources being protected and to prevent user-level or application-level exploits from being able to circumvent security. When security information is placed on every object on the system, and the kernel itself makes security decisions, the system is more secure than any combination of user-level or application-level security.

PitBull adds security to file system objects and to processes. It also replaces the standard UID 0 checks with more specific and more targeted privilege checks. And PitBull products install directly onto an operational platform without requiring the removal of the commercial OS and COTS applications. Users installing PitBull enjoy the benefits of retaining the look and feel of a known commercial product with only the requirement to learn the incremental enhanced security capabilities over and above their native OS.

System users installing PitBull in accordance with the PitBull .comPack Installation Guide and user training should experience the following benefits:

- COTS-like software installation over an existing configuration.
- Continued operation of those existing software applications conforming to standard Solaris or AIX APIs.
- Immediately improved security and assignment of default sensitivity labels associated with existing applications.
- The ability to increase the security of commercial software applications with the `tracepv` utility, through which the minimum privileges required for operation can be determined.

Trusted operating systems, though secure, are of limited utility unless the software applications required by the users operate effectively and efficiently. 1st and 2nd Generation trusted products were often characterized by requirements for extensive integration activities and therefore had limited utility due to ongoing requirements to customize new applications. Standard application interfaces were typically modified to support security requirements, thus negating their utility. Extensive integration and maintenance activities often made Trusted Systems too costly to implement.

Solaris 2.5.1 - 2.8 and AIX 4.3.3 systems upgraded with Argus enhanced security products deliver 100% compatibility with the commercial Solaris and AIX application-programming interfaces (APIs). Any application that runs on commercial Solaris or AIX will also run on the system with Argus enhancements, without modifying the existing application. (One exception is in the instance of applications which modify file systems or the operating system kernel; these kinds of products will not be able to operate without modification on an Argus system.) Furthermore, all applications residing on a system prior to the Argus installation will be labeled with SLs according to system defaults. Argus extends the Solaris and AIX APIs to support additional security functionality. This allows customers to develop their own secure applications using the Solaris API and Argus extensions. Argus extensions are documented in the Argus Software Developer’s Kit.

Unfortunately, not all commercial off-the-shelf applications are “well behaved;” i.e., they perform various functions that violate the security policy of the operating system. Examples are applications that require that they execute as the superuser or root account. Argus has provided tools and features to run such programs on an Argus system, while still minimizing threats to system security:

- A fine-grained set of privileges that can be used by the ISSO to assign privileges to an application on a case-by-case basis so that the application functions correctly on the system in a minimal-risk, secure manner. The principle of least privilege requires that application programs be assigned only the minimal set of privileges necessary to allow the application program's performance.
- A superuser emulation feature to allow those applications that assume the superuser status to execute within the control of the trusted computing base without superuser status. Programs run in this method will have control of standard UNIX security mechanisms, but will not be able to override the PitBull security mechanisms.

Building on the foundation: PitBull .comPack

In simplest terms, PitBull Foundation is about separating elements from each other and making access between areas more difficult. PitBull .comPack is about building specific connections between areas that need to communicate securely.

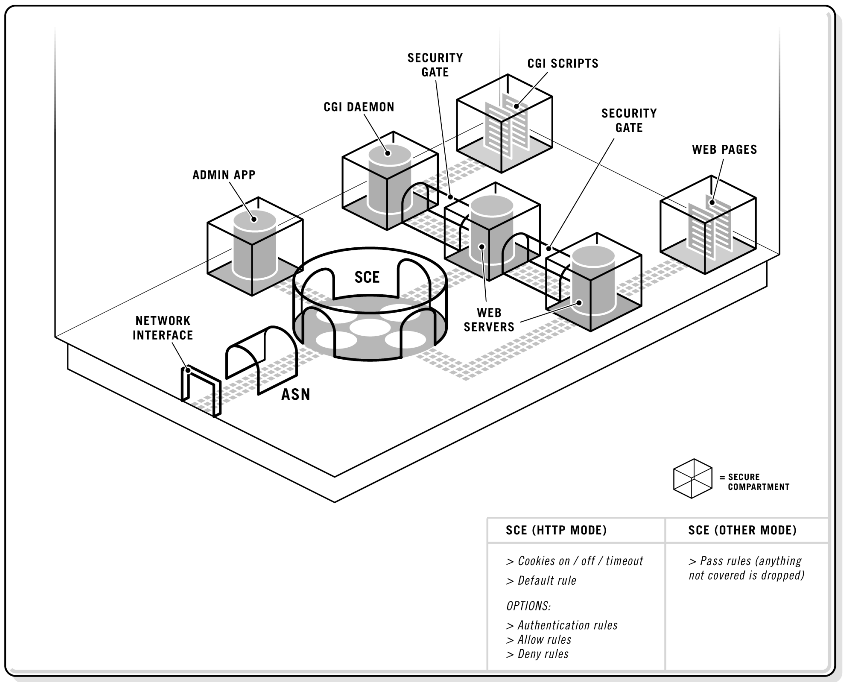
In addition to the security features provided by PitBull Foundation, components included within PitBull .comPack offer a solutions-oriented suite which protects the system from intruders and errant software. These highly specialized software modules separate functional components in order to minimize the chances of attack. Because application functionality is separated from important data, there is no way one application can influence or manipulate the operations of a critical process. There are seven distinct modules:

- The Secure Communications Enforcer (SCE) intercepts selected communications from a network and redirects them to daemons. Redirection decisions are based on the security level of the incoming communication, the URL specified in the request, and sets of rules determined by system administrators. With the SCE, only properly authenticated clients at the proper security levels can access secure back-end applications.
- The Security Gate acts as a secure, controlled channel between two programs operating at different security levels, allowing them to communicate within predefined limits and preventing front-end exploits from being able to access unauthorized areas behind the security gate.
- The Secure Authentication Module places the authentication responsibilities of the web server into its own compartment, in order to restrict access to the sensitive information typically involved in user authentication.
- The Secure CGI Module isolates the CGI functionality of a web server from the web server itself. With the Secure CGI Module, the CGI executables are run at a different sensitivity label and from a different location than the rest of the server, to prevent CGI-based exploits from affecting any other files.
- The Secure Program Launcher (SPL) allows specific users without powerful authorizations to execute programs that operate at a high level of security, but only in a predefined manner. This increases security by allowing users to perform pre-configured tasks without permitting them direct control of greater privileges than the users are authorized to have.
- Trusted SSH, Argus's implementation of F-secure's SSH (Secure Shell), permits remote trusted administration of a PitBull system through key exchange and encryption. The Argus-enhanced version also permits administrators to specify a sensitivity label at login.
- IPSec, an encrypted IP protocol, enhances the PitBull .comPack suite of products by providing VPN-like capabilities through the use of both keys and encryption. This provides additional security in the areas of authentication and confidentiality.

These components are available as a fully integrated product suite or individually as separate modules to provide a secure platform for web-based services or hosting applications. Users install PitBull .comPack and immediately begin taking advantage of its security features. Rather than requiring administrators to replace the commercial operating system altogether (as with all other trusted operating systems), both PitBull Foundation and PitBull.comPack enhancements are installed as system upgrades.

In addition to improving overall system security, PitBull technology also enhances other traditional security systems such as firewalls and authentication. Without the PitBull foundation of security, these other mechanisms are still vulnerable to any exploit in any program which permits access to a root-owned process. A PitBull system's elimination of root power, separation of privileges, and compartmentalization of applications provides the critical layer of security required to ensure the integrity and confidentiality of data and applications at all levels of operation.

Secure Communications Enforcer



SCE (HTTP MODE)	SCE (OTHER MODE)
> Cookies on / off / timeout *> Default rule* *OPTIONS:* *> Authentication rules* *> Allow rules* *> Deny rules*	*> Pass rules (anything not covered is dropped)*

FIGURE 2. **Secure Communications Enforcer (SCE)**

The Secure Communications Enforcer (SCE) would normally be set up on the front-end web server, although it can also be set up with other programs. It can handle requests for compartments on a one-to-one, one-to-many, many-to-many and many-to-one basis. All communications can be directed through the SCE on the system to control remote access to sensitive system services and data. The SCE was formerly known as the Upgrade/ Downgrade Enforcer (UDE), and is referred to in that manner in the documentation, commands, and software.

The SCE intercepts all communications from the network on a specific port and redirects them to selected servers and daemons. Although web servers can provide their own identification and authentication schemes, an exploit of the authentication mechanism would render all data contained within the web server vulnerable to unauthorized access. To prevent this, the SCE uses Sensitivity Labels (SLs) assigned to each network connection by Advanced Secure Networking (ASN) to distinguish among clients and to forward each request to the appropriate back-end web server.

When a client makes a request, incoming connections can be sent to the SCE. When the connection is received by ASN, the data stream is assigned an SL according to specific rules set forth in the ASN configuration. The Secure Communications Enforcer reads the SL (provided by ASN) and the URL of the request, then applies rules in its configuration file to determine which web server each client can access. In this manner, trusted clients, such as internal users, can be assigned labels that the SCE redirects to a sensitive web server, while Internet users are only allowed to access a web server designed to meet public needs.

Before sending a request to a web server, the SCE re-labels that request, according to rules in its configuration file. The rules are set up so that the new SL of the packet matches the SL of the target server, so that the server can read the packet.

Security Gate

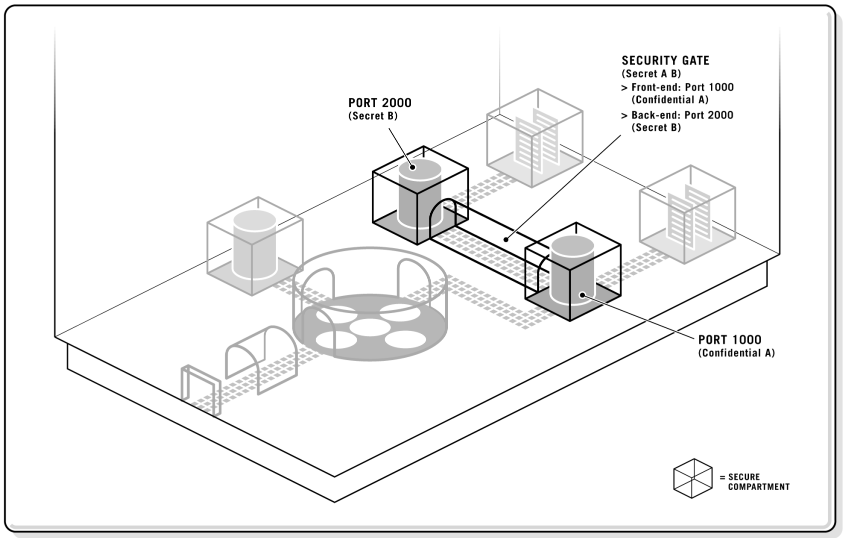


FIGURE 3. **Security Gate**

The Argus Security Gate is a small trusted program that allows two processes with disjoint SLs to communicate with each other. Under ordinary circumstances, processes with disjoint SLs (SLs which have no relationship, so that neither one dominates the other) are not permitted to communicate. The Security Gate is typically used to allow a front-end client to communicate with a back-end server that runs at a different sensitivity level. In this way, the Security Gate acts as a secure controlled channel between the two programs, allowing them to communicate with each other but not allowing them to access each other's files directly. Since a typical installation has several back-end servers, there are often several Security Gates running simultaneously, one for each back-end server.

Once the connection to the back-end server has been established, the Security Gate passes information between the front-end client and the back-end server. The SL for incoming data is compared to the configuration settings for the Security Gate and if the SLs do not match, that data connection is dropped.

The Security Gate does not modify or examine the data that it passes between the front-end client and the back-end server, allowing the Security Gate to mediate many different user-level protocols. The Security Gate supports several Stream and Datagram protocol types. The supported Stream protocols are TCP and UNIX Domain protocols, and the supported Datagram protocol is UDP. Datagram broadcasting and the Datagram UNIX Domain protocols are currently unsupported.

The Security Gate differs from other PitBull .comPack applications in that it supports UDP sockets in addition to TCP and UNIX Domain sockets.

Secure Authentication Module

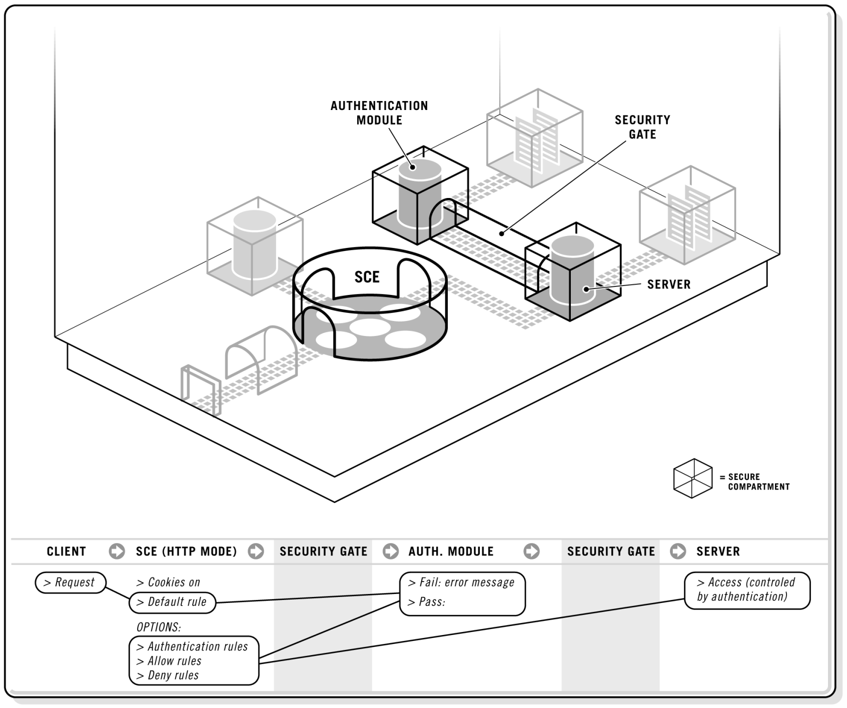


FIGURE 4. **Secure Authentication Module**

The purpose of the Argus Secure Authentication Module is to place the authentication responsibilities of the web server into its own compartment to restrict access to sensitive information involved in user authentication. There are many different ways to authenticate users; the most common is identification and password, but many systems are moving to PKI, two-factor authentication and biometrics. These programs would be running on the same machine as the Secure Authentication Module, but they can now be compartmentalized to enhance the security of the application itself. The Secure Authentication Module was formerly known as the Authentication Daemon (`authd`), and is referred to in that manner in the documentation, commands, and software.

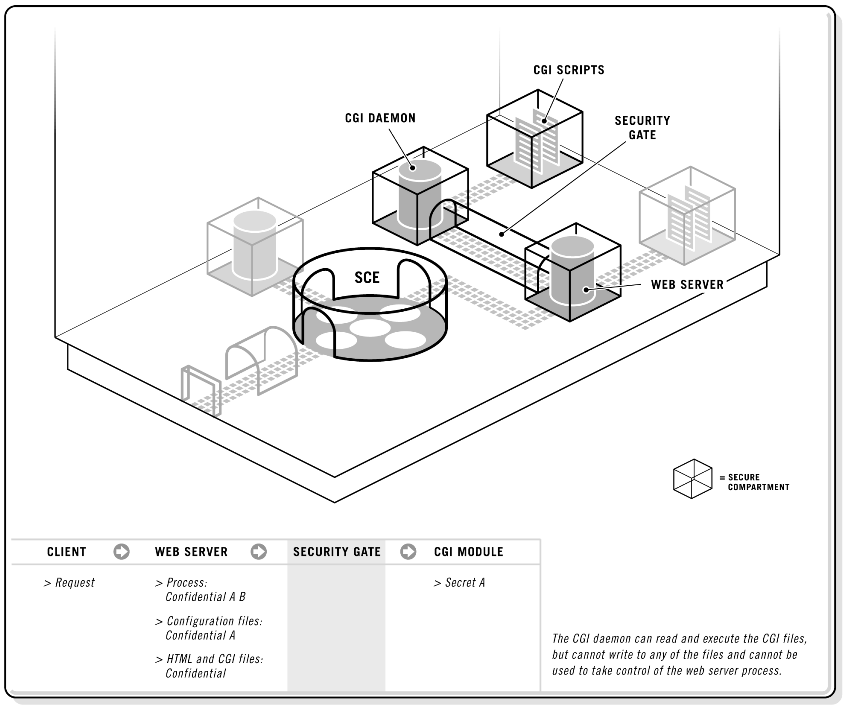
When started, `authd` automatically detaches from the controlling terminal and puts itself in the background. Afterwards, it listens on a socket specified in the configuration file and waits for requests from a web server. When `authd` receives a request, it executes an authentication script. If the return value of the script is zero, `authd` creates an HTTP header returned to the client browser. The header causes the client browser to set a cookie in its memory, and the client browser sends that cookie to the SCE with each subsequent packet. The cookie is a hashed value generated with the RSA Data Security, Inc. MD5 Message-Digest Algorithm.

The SCE must be used in conjunction with the Secure Authentication Module, and the authd configuration file must contain the URL of the SCE for a cookie to be directed from the client back to the SCE. When the SCE receives the cookie from the client, it compares the cookie to the servers listed in its AUTH rules. If matched, it promotes the client to the specified back-end web server. The SCE does this for each packet until the cookie times out. For authentication to work, the SCE and authd must have corresponding AUTH rules in both of their configuration files.

If the authentication script returns any value other than zero, the user receives an error page specified in the configuration file. The cookie is never stored in permanent memory on the client machine. Once a client is promoted to a different server, clicking the browser's BACK key will produce an error message.

The Secure Authentication Module interacts with the web server through the use of Argus-provided plug-in modules to the web server. Currently, Argus supports Netscape Enterprise Server 3.6.2, 4.0, and 4.1, and Apache 1.3.9. The Secure Authentication Module must be linked into the web server to which the SCE sends authentication requests. The web server's authentication module intercepts all requests that end with the extension configured in the MIME.types file, and forwards them on to a Security Gate, which in turn redirects the request to authd.

Secure CGI Module



CLIENT	WEB SERVER	SECURITY GATE	CGI MODULE
> Request	> Process: Confidential A B > Configuration files: Confidential A > HTML and CGI files: Confidential		> Secret A

The CGI daemon can read and execute the CGI files, but cannot write to any of the files and cannot be used to take control of the web server process.

FIGURE 5. **Secure CGI Module**

The Secure Common Gateway Interface (CGI) Module serves as a means to separate the CGI functionality of a web server from the web server itself. By using a Security Gate to broker communications between the Secure CGI Module and the web server, both components can be isolated to prevent and limit losses in the event of a web server compromise. Compartmentalization using mandatory access controls (MAC) ensures that exploits on the web server cannot affect secure operations of CGI scripts and vice versa. The Secure CGI Module was formerly known as the CGI Daemon (`cgid`), and is referred to in that manner in the documentation, commands, and software.

When started, `cgid` automatically detaches from the controlling terminal and puts itself into the background. Once it has done so, it listens on a port as specified in the configuration file and waits for requests from a web server. When `cgid` receives a request, it sets environment variables according to directives in the configuration file, and invokes the CGI script. The script's output is then returned to the web server.

The Secure CGI Module interacts with the web server through the use of Argus-provided plug-in modules to the web server. The module needs to have a Security Gate to be able to communicate with `cgid`, which runs in a disjoint compartment. Currently, Argus supports Netscape Enterprise Server 3.6.2, 4.0, and 4.1, and Apache 1.3.9. The web server's plugin module intercepts all CGI requests and forwards them to the Security Gate, which in turn redirects the request to `cgid`. After execution of the script, `cgid` returns the script output to the Security Gate, which sends it to modules within the web server and then back to the client's browser.

Secure Program Launcher

The purpose of the Secure Program Launcher (SPL) is to allow system administrators to create startup and shutdown scripts for programs that need to execute at a specific sensitivity level, without requiring those scripts to be run by users with powerful authorizations (such as ISSO). The SPL was formerly known as Execenv, and is referred to in that manner in the documentation, commands, and software.

SPL uses a database to determine how to execute a program. Each entry in the SPL database, `/etc/security/execenv.conf`, informs SPL how to execute one program. The SPL command line argument is a tag that tells SPL which entry in the database to use.

The SPL program is capable of running programs that require interactive input, even if the program turns off terminal echoing. This allows SPL to execute programs that require passwords.

Many measures have been taken to avoid the possibility of exploitation of an SPL-initiated process, including:

- Privilege stripping: The SPL removes any privileges inherited from the process which called it, so that the process it creates runs with only its own innate privileges.
- Group stripping: The SPL empties the supplementary group list when it executes a program so that the program runs only with the primary group of the user specified in the execenv.conf file.
- Environment-free execution: The SPL executes programs with no environment to avoid shell scripting exploits.
- Pseudo-terminals: By default, the SPL sets up a pseudo terminal for each spawned process to allow the user's terminal settings to remain unchanged.

Example of SPL Usage

Suppose that a web server in a particular installation should run at the sensitivity level CONFIDENTIAL WEBSERVER. The system administrator could decide to write a shell script to launch the web server:

```
#!/bin/sh
/tbin/setsl -a "CONFIDENTIAL WEBSERVER" $$
/tbin/setpv a "" $$
/opt/webserver/start_web_server
```

The problem with this script is that the `setsl` and `setpv` commands require very powerful authorizations. Without the Secure Program Launcher, the web administrator would need SETSL and SETPV authorizations, which would also permit the web administrator (or anyone who hacked his account) to use those commands on other programs or files than the script which starts the web server.

If the site has a dedicated web administrator who maintains the web server and nothing else, the system administrator can give the web administrator the SPL authorization (EXECENV) to be able to fix this problem. The system administrator can create an entry in the execenv.conf database that executes the web server at the correct sensitivity level and with the privileges the web server requires to run. When the web administrator executes the SPL program, he can start and stop the web server with the chosen settings, without being permitted to change his own process's settings to match those of the server or other processes on the system.

Remote Administration with Trusted SSH

Administration of web and transaction servers represents a significant task. Additionally, it raises security challenges since administrators of standard systems often have access to virtually all system resources. Remote administration of standard systems therefore can present a significant security risk should an administrator's authentication become compromised. PitBull .comPack provides a highly secure and configurable mechanism for providing and securing remote administration of PitBull systems.

In PitBull .comPack products sold in the United States, secure remote administration may be performed through Data Fellows' F-Secure SSH Server (SSH). SSH is a client/server architecture that encrypts communications between machines, which is provided as part of the PitBull .comPack installation. This method of communication eliminates the risk of other users intercepting network administration data prior to its delivery to the destination system.

Because of the secure nature of the SSH architecture, certain privileges are granted to the secure shell daemon (`sshd`) that allow it to have more flexibility on a PitBull .comPack system than any other form of network communications. As a trusted application, `sshd` is not subject to the Advanced Secure Networking (ASN) restrictions set forth in the ASN configuration files. Ordinarily, any network connection is subject to the most restrictive rule for that client, either given to the network interface on which the connection was made or specifically stated for the client machine. These rules define the range of sensitivity labels (SLs) that are allowed to enter or exit the machine. `sshd`, however, is not subject to this restriction because it can deny connections other than those specified in its configuration file. More importantly, it uses RSA public/private key authentication, which prevents "spoofing" a network packet and fooling the server into believing that the client is a trusted host when it is not. Instead, a user on a trusted remote machine is allowed to connect at any sensitivity level that is within the user's clearance, as defined in the server's `/etc/security/clear` file. With these capabilities, `sshd` allows connections only from trusted hosts, thus ensuring that information entering or leaving the machine can only be viewed by authorized users.

The sshd program available from Argus has been modified to allow clients to connect at specified sensitivity levels. Any `ssh` client can communicate with the modified `sshd` and specify the effective SL of the connection.

Along with the variation of command line arguments used with the `ssh` client, `sshd` also has noticeable differences to facilitate its integration to the PitBull .comPack system. Typically, the `ssh` daemon has a "set user id" (`setuid`) of zero, which causes it to run with the same privileges as root on a non-PitBull .comPack system. On the PitBull .comPack system, however, `sshd` is configured to run as user `sshd` with an SL of SYSTEM_HIGH. The `ssh` daemon also operates with a least privilege concept. Instead of assigning the daemon superuser rights as with non-PitBull .comPack systems, `sshd` is only granted enough privileges to function properly. The daemon drops any extra privileges immediately after the connection from the remote host has been made.

Using TSSH allows for the secure administration of a PitBull .comPack server from any trusted host with a significantly increased level of security.

IPSec

PitBull and PitBull .comPack integrate the IPSec protocol into the IPS security solution to consistently provide a high level of security not only on the host, but also along the wire and network as well. Using modern cryptographic methods, IPSec uses privacy and authentication services to provide a standard, secure way of communication with the TCP/IP protocol. IPSec can protect any IP-based service or application.

IPSec covers three functional areas of security: authentication, confidentiality, and key management. It is useful for secure branch office connectivity over the Internet, secure remote access over the Internet, establishing extranet and Intranet connectivity with partners, and enhancing e-commerce security.

IPSec provides security services at the IP layer by giving a system the ability to configure the handling of traffic, providing access control, connectionless integrity, data origin authentication, rejection of replayed packets, confidentiality, and limited traffic flow confidentiality.

Argus implements IPSec as a way to provide VPN-like capabilities with the IPS to present a rock solid foundation for the enterprise.

Common architectures: putting it into practice

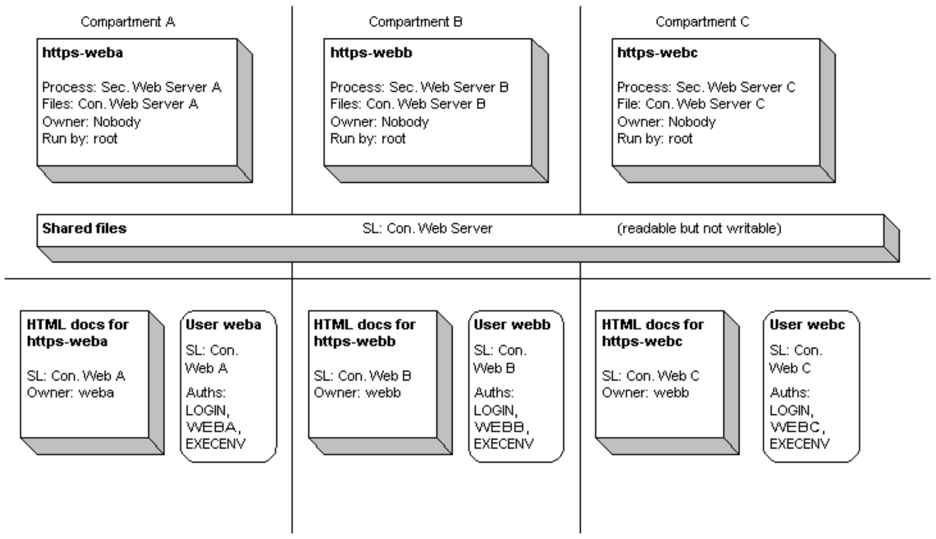
Our customers include some of the world's biggest names in Internet commerce, finance, medical records, and telecommunications; some sample architectures are described below.

Internet service providers

PitBull Foundation and the Secure Program Launcher module of PitBull .comPack provide a secure base for Internet service provider (ISP) businesses, which must provide access to a wide range of users without permitting those users to affect their virtual neighbors' sites. With the compartmentalization abilities of PitBull Foundation and the privilege-stripping secure startup of the SPL, hundreds of different web sites can be hosted on the same machine or group of machines. Each site owner can be permitted as much control of their own web server as the ISP administrator wishes to grant, whether that means limiting the users to modifying HTML files or whether it means permitting them to modify their server files and restart their servers. And each site owner can be effortlessly prevented from writing to, restarting, or even viewing the other site owners' servers and files.

Foundation and SPL

Using an example with three site owners, Adam, Beth, and Charlie:



The ISP administrator installs three web servers and secures them with disjoint sensitivity labels. Adam's web server runs at an SL of Confidential Web Server A (three distinct compartments: Web, Server, and A). Similarly, Beth's web server runs at Confidential Web Server B, and Charlie's web server runs at Confidential Web Server C. In this example, the server executable files are stored at a different SL than both the web server and the HTML files, to prevent either the users or hackers who exploit the server process from modifying them. However, files that are shared among the three servers (such as a frame identifying the ISP and its associated graphics) are stored at a level which all three server processes dominate: Confidential Web Server. (Neither users nor hackers will be able to modify these files either.)

Each of the users are given a login name with a single-level clearance. Adam's clearance is Confidential Web A, Beth's is Confidential Web B, and Charlie's is Confidential Web C. In addition, each user is given the LOGIN and EXECENV authorizations. The third authorization in each user's list belongs to that user alone. This authorization is used in conjunction with the SPL to permit this user to restart his or her server if needed, and as an access authorization on any server configuration files which the ISP administrator decides the user should be allowed to modify. (Both the HTML files and those selected configuration files should be stored at the user's single level. In addition, any directories these users write in should have the minimum and maximum SLs both set to the user's level, so that a server exploit cannot be used to write new files at the server's SL in these directories.)

The result of this is that the users can restart their own web servers at the correct SL for the web server process without being able to change his/her own SL to that of the process or perform any other functions at that SL. Each user can also edit his or her own web server configuration files and HTML files. However, no user can perform these functions on anyone's web server but his or her own.

TSSH

To prevent password-based exploits, the ISP administrator can require connections to be made with TSSH, using RSA keys. Each user is given a unique key to store on the client machine from which he or she will connect to the server machine. Then the server machine can be configured to refuse password-based connections to that account. Provided that the user does not distribute the access key and the passphrase which he or she uses to enable that key for use on the client machine, the account will be secure from password-based attacks and all traffic between that client and the server will be encrypted.

Secure CGI Module and Security Gate

At an ISP which would like to offer its customers secure CGI services without either needing a time-consuming and expensive professional review of each CGI script submitted by their customers or a separate machine for each customer, the Secure CGI Module is invaluable. This module permits the CGI part of the web server to be isolated from both the server and the CGI and HTML files, so that a CGI exploit can't be used to modify any other part of the server -- including its own files. A Security Gate between the web server and the Secure CGI Module permits specific point-to-point access between the disparate SLs.

To build on the example above, rather than allowing the users' web servers to interpret the CGI directly, the server configurations would be modified to use the server's designated Secure CGI Module via a Security Gate. Each Secure CGI Module would run at a different SL than both the server and the documents. To use server A as an example, the Secure CGI Module could run at a SL of Confidential Web A CGI, and the Security Gate brokering communication between the server and the CGI Module could run at Confidential Web Server A CGI (in order for its label to dominate but not equal both the web server and the CGI Module). The CGI scripts themselves would be stored in a designated directory set in the CGI Module's configuration. The scripts would be stored at the same SL as the other HTML files (Confidential Web A). This way, the user could edit them as needed; but any exploits in the CGI script would leave a hacker with a process running at Confidential Web A CGI, which would not be able to modify any of the files.

Secure transactions: commerce, banking, corporate intranets

Although securing web sites is a common use of the PitBull .comPack modules, they can be used with applications other than web servers. Security Gates can be set up between any two applications or services which use ports for communication. And any number of Security Gates can send information to the same back-end port.

Continuing with the model above, suppose that the three web sites all need to communicate with the same back-end database. The database's SL is Secret Database. The CGI scripts handled by the Secure CGI Module handle modifications to and searches of the database. So, by placing a Security Gate between each web server and the database, any user who wishes to communicate with the database must do so through the medium of the approved pages on the web server -- which can't be modified because there is no way for a hacker who compromises the server process to lower the SL of that process to the level of the CGI scripts and HTML pages.

Secure Communications Enforcer

In a large business, that type of setup would require a large number of Security Gates in use, which is not an efficient use of port numbers. The Secure Communications Enforcer is intended to receive requests from multiple SLs to multiple SLs and redirect them appropriately, based on site-defined allow and deny rules. A single instance of the SCE could replace the multiple Security Gates used for database access above, and could handle server requests for communications with additional back-end applications at the same time. Because the SCE can communicate in either HTTP mode or OTHER mode, it can be used in conjunction with web servers or with other applications as needed.

Secure Authentication Module

The SCE can also be placed in front of web servers, as part of a user authentication system in conjunction with the Secure Authentication Module and ASN. The Secure Authentication Module requires the SCE and an authentication executable of your choice; like PAM, the Secure Authentication Module is highly adaptable to site-specific needs.

In a typical configuration, the SCE would be placed on port 80 to receive all requests which are targeted for the main web server and arrive at specific SLs. (If desired, requests which arrive at other SLs can be denied.) A form on the index page would allow users to authenticate themselves (via the Secure Authentication Module and a Security Gate, running at different SLs in a manner similar to the Secure CGI Module description above). The Secure Authentication Module could hold authentication rules for each of the servers in the example, and with the SCE, connections which have passed the Secure Authentication Module would be automatically redirected to the server for which the user is authenticated.

IPSec

IPSec's purpose is to provide data-level or packet-level encryption for networking traffic. At the data level, the information contained within the networking packet is encrypted during transmission but the IP header and other networking information is left unencrypted. At the packet level, the entire networking packet is encrypted and IPSec inserts its own header and routing information. The advantage to the latter approach is that the data type and eventual target cannot be read from the packet as it travels. Using this encrypted network communication with RSA key authentication or other strong authentication methods provides the ability to create Virtual Private Networks (VPNs), which are a vital security component for Internet-based "Just In Time" business models.

Evaluation and Certification

As an internationally renowned provider of security solutions, Argus Systems Group fully understands the importance of independent product evaluation and validation. As such it has participated in both the TCSEC and ITSEC processes, and is currently participating in a Common Criteria evaluation process. PitBull security technology has been ITSEC-certified at the F/E3 B1 (Labeled Security Protection) level of assurance, although it contains functionality approaching the B2 level. This certification, which includes the networking components, is higher than for any other commercially available products in PitBull .comPack's class, thus ensuring that its users have the most advanced secure system product available today.

PitBull Foundation, the foundation security product upon which PitBull .comPack is built, is currently successfully undergoing testing in the National Security Agency's sponsored "Trust Technology Assessment Program", and our Solaris-based product will soon complete testing for EAL 3 certification, followed by EAL 4 certification.

For more information

For more information about Argus Systems Group and its market-leading suite of trusted operating systems technology:

- Email: info@argus-systems.com
- Website: www.argus-systems.com
- Fax: (217) 355-1433
- Address:

  Argus Systems Group, Inc.
  1809 Woodfield Drive
  Savoy, IL 61874
  USA
- United States: (217) 355-6308
  France: +33 14759 2348
  Germany: +49 228 98356 0
  Switzerland: +41 61 228 9200
  UK: +44 1494 430 300