Interested in learning
more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Secure OS Environments for Linux

In this paper I make a review of the main set of tools and resources available for Linux system administrators willing to build an operating system with enhanced security features that allow applications to run securely in a network accessible from the Internet. I have summarized the state of the art in this subject by offering an overview of the tools, compiling the most useful references and classifying them accordingly. The ultimate goal of the paper is to make more affordable the initial work for anyone interested ...

Copyright SANS Institute
Author Retains Full Rights

AD

Beating the IPS?

STONESOFT
Can't be Beat

Learn
More

Secure OS Environments for Linux

By
Pedro A. Luz-Romero
April 14, 2003

GIAC Security Essentials Certification (GSEC)
Practical Assignment – Version 1.4b
Option 1 – Research on Topics in Information Security

© SANS Institute 2003, Author retains full rights

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

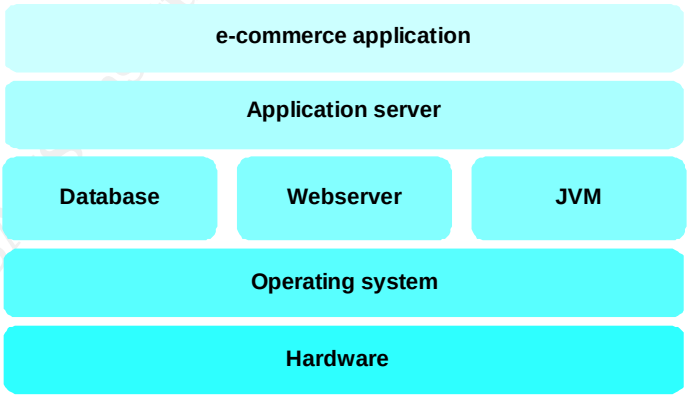
Abstract

In this paper I make a review of the main set of tools and resources available for Linux system administrators willing to build an operating system with enhanced security features that allow applications to run securely in a network accessible from the Internet. I have summarized the state of the art in this subject by offering an overview of the tools, compiling the most useful references and classifying them accordingly. The ultimate goal of the paper is to make more affordable the initial work for anyone interested in this topic.

Software architecture complexity

Currently firewalls and other access control methods are used as gatekeepers to control the access from internal and external sources. However, some ports must remain open in firewalls to enable the application server to be accessed. The protection provided by firewalls is not enough over these open ports because their work is to block some ports and to forward IP packets in others, in a state-based way in some cases, so hosts receiving requests are responsible for enforcing security measures needed. On the other hand, in most situations, hazards are not originated by Internet intruders, but are the result of system abuses from authorized internal users. In the latter case firewalls cannot offer any protection so hosts must protect themselves against this threat.

At the same time, current software systems are becoming more complex. As we can see in the figure, each box represents a software subsystem probably developed independently by different suppliers and later integrated by another company. Therefore, in such systems we have to ensure security both in each box and at the integration process. Basically, we have to trust that everybody made his work with security in mind.



One expects that in case any software subsystem has any kind of security defect, the security of the entire host or others host in your intranet should not be compromised. You can prevent this by limiting communications and relationships to those that are absolutely necessary. In fact, you will need to look carefully at two issues. First, and this is what traditional firewall security is

© SANS Institute 2003. As part of GIAC practical repository. Author retains full rights.
© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

all about, to define services the host provides to internal and external machines and determine, on a service-by-service basis, what communication channels hosts really need to have. Secondly, you must also think about the relationships between processes and objects inside the host. Once you have made these decisions, you can use a variety of mechanisms to enforce them.

Through this paper I will describe the most representative mechanisms to restrict the relationships between processes and objects inside the host, given their relevance in today's computer security field. The best place to implement these security mechanisms is the OS layer. After all, this layer is the base for all the other subsystems and should be the one on which we have more control. I have chosen to focus in mechanisms available for the operating system Linux, given its growing relevance in computer user community.

In next section I will review different Linux distributions currently available and will describe their built-in security features. It is possible that some of them match your requirements straightforwardly. If that is not the case, the following section offers general guidelines to design an ad-hoc customization process to suit your security needs. Finally, there are some cases where very specific features are need. I will describe in the last section several modifications that can be applied to the Linux kernel in order to add, enhance or modify some security characteristics.

Linux Distributions Maze

As mentioned in the previous section, the operating system is the best place to implement security policies. In the Linux world it is not difficult to find hundreds of different distributions, each having its pros and cons. It is not the objective of this paper to compare them but rather to classify them according to the way they handle computer security. Among all of them, it is possible to distinguish clearly two relevant types of distributions for the context of this paper: general-purpose distributions and "secure-featured" distributions. To set up a secure operating system environment we could use any distribution of the two types. A general-purpose one should follow a hardening process. On the other hand, a secure-featured one would meet our requirements without needing a lot of customization.

General purpose distributions

We can classify in this category the majority of the most popular Linux distributions. Basically, each of them consists of a kernel plus a set of diverse utilities and application packages. These distributions offer the added value of a simplified and homogeneous installation and administration environment in such a way these distributions adopt a unique personality. Red Hat[24], Mandrake[19] or SuSE[27] are all well-know general-purpose distributions.

© SANS Institute 2003. As part of GIAC practical repository. Author retains full rights.
© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Linux security-featured distributions

These distributions offer to system administrators some added value from the point of view of securing the operating system. It is easy to distinguish among the following three types of distributions.

Appliances Linux

These are reduced Linux distributions. The aim of these distributions is to enable small hardware resources boxes with a minimal kernel and configuration to perform a very specific task. Normally, they present a subset of the followings features

- They boot from a removable storage, possibly write-protected (CD, EPROM, floppy) avoiding an intruder could write the file system.
- They are executed in a ram disk, they normally don't need/use a hard disk, so zero administration can be easily fulfilled.
- These distributions allow the configuration to be stored either in a ram disk or a write-protected disk, preventing an intruder from modifying it.
- These boxes usually perform a network task, as firewall or router, usually based on Linux Router Project (LRP).

Although they can be useful in certain cases, these distributions lack the necessary flexibility to execute the majority of modern applications. Due to its simplicity, it is difficult to administer its configuration and to carry out diagnoses on them. Some examples of these distributions are Frazierwall Linux[8], Fli4L[20], Floppyfw[35] or Devil-Linux[36].

Secure corporate Internet services

These distributions are designed for small offices in which a product is needed to enable quickly a series of basic Internet services to support business processes. Staff, which is usually not security-aware, implements normally these services. They share several of the following features

- Secured operating systems with a specific security policy applied.
- Internet services installation using secure configurations by default with simple administration environments for every service, avoiding typical misconfigurations in plain text files.
- Host based firewalls and built-in intrusion detectors.
- Remote access products with encryption capabilities.
- Graphical tools to custom easily security policies.

Frequently, this kind of distributions shares some features with the following group. However, I still put them in this category if their main objective is not security but to provide enterprise services. Astaro Security Linux[2], EnGarde Secure Linux[9], Firegate Server[30] and Immunix OS[33] are distributions to classify in this category.

© SANS Institute 2003. As part of GIAC practical repository. Author retains full rights.
© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Secure out-of-the-box

These distributions are built on top of a Linux kernel and have been added security extensions. Normally, they come from the factory with a modified kernel and a default configuration with advanced security settings. An example of those basic security settings is the least privilege principle. These distributions usually include the following features,

- Limited access to system data and resources. Controls may be set on all potential interactions with programs, file access, and utilities on a user-by-user basis.
- Eliminate super user. Dividing super-user functions into multiple roles makes penetration far more difficult.
- Improved security auditing. Actions that may affect security or sensitive files can be monitored. To detect suspicious actions, administrators may generate usage reports by user, group, file, data, or time.
- Provide a "trusted" path that protects entered data. This is particularly important for passwords, which may also be protected by using and enhanced encryption algorithm.
- Use of sophisticated authentication methods or a modified PAM, not only for system access but also for subject-to-object general access.

The fundamental advantage of this type of distributions over the rest is that they have been designed with security in mind. Some of these distributions are so focused on security that neglect the rest of operating system aspects. Others are research prototypes developed by universities without any commercial aim. These facts make them neither attractive nor widely used for business purposes. Some distributions of this type are Kaladix Linux[18], SE Linux[21], Trustix AS[28] or CAEN Linux[29].

Building a Bastion Host

Where possible, installing a security-featured distribution should be sought. If any requirement, like support for the software of a third party vendor, leads us to the installation of a general purpose distribution, we still we can endow this distribution with a certain security level. To this aim, we will customize the distribution for our application by applying a hardening process. This process can also be applied if we install a security-featured distribution since probably this one does not fulfill all the needed requirements.

Basic philosophy …

There are two basic principles for securing a host:

Keep it simple

The simpler your host is, the easier it is to secure. Any service the host offers could have software bugs or configuration errors on it, and any bugs or errors may lead to security problems. Therefore, you want the host to do as little as possible. It should provide the smallest set of

© SANS Institute 2003. As part of GIAC practical repository. Author retains full rights.
© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

services with the least privileges it possibly can, while still fulfilling its role.

Be prepared for the bastion host to be compromised

Despite your best efforts to ensure the security of the host, break-ins can occur. Don't be naive about it. Only by anticipating the worst case, and planning for it, will you be most likely to avert it. Always keep the question, "What if the host is compromised?" in the back of your mind as you go through the steps of securing the machine and the rest of the network [5].

… but multiple tools

Now that you have figured out what you want your bastion host to do, you need to actually build the host. In order to do that, follow any of the useful guides or tools available in the Internet. Basically, these references[4][15][16] recommend following these steps:

- Install the minimum software needed.
- Secure the machine.
- Disable all non-required services.
- Install the business-logic applications host must provide.
- Install patches that match your platform.
- Run a security audit to establish a baseline.
- Connect the machine to the network it will be used on.

You should be very careful to make sure the machine is not accessible from the Internet until each of the previous steps has been followed. If the bastion host is vulnerable to the Internet while it is being built, it may become an attack enabler instead of a defense mechanism. An intruder who gets in before you have run the baseline audit will be difficult to detect and will be well-positioned to read all of your traffic to and from the Internet. Cases have been reported where machines have been broken into within minutes of first being connected to the Internet; while rare, they may happen.

Secure system libraries

The weaker link in the process previously described is the business-logic applications. It is well known that developer teams are more worried about fulfilling customer expectations and time-to-target objectives than about designing a secure application. To avoid this problem system administrators should, whenever possible, enforce the use of secured system libraries. This product is highly recommended to prevent, detect and handle buffer overflow attacks.

Currently, there are two kinds of secure system libraries. The simplest one is a compiler extension that generates auto-checking stack bound code. Positive of this approach is its ease of use but negative is that the source code of the application must be available to be recompiled. A well-known implementation of this technology is Stackguard[34]. The checking mechanism it uses is based on the generation of an additional field that is placed next to the return

© SANS Institute 2003. As part of GIAC practical repository. Author retains full rights.
© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

address every time a function is called. When function returns this field is checked and if it has been modified a stack smashing attack has been attempted. In this case the application write an alert into syslog, and then halts. The fact that checked fields are usually generated randomly and the fact that this process is performed every time a function is called could affect application performance seriously so it should be carefully studied before its using in production environments.

On the other hand, there are other solutions that do not need the source code to be available[3]. These libraries try to avoid buffer overflows for well-known vulnerable C library functions that handle buffers as strcpy, gets, fscanf, sprintf, … These solutions are implemented as dynamic loaded system libraries that contain a alternative version of each vulnerable function that perform its traditional work in a more secure way using techniques to detect buffer overflows. Environments variables should be setup to intercept calls to these well-known vulnerable library functions so they are performed by dynamically linked applications instead of by the traditional ones. This alternative has smaller performance impact because security checks are only performed when a modified function is called. On the other hand it only offers protection if the application is written using these functions.

Kernel hardening

Nobody can assure a bastion host will never be the target of a security incident. Every day, new vulnerabilities are discovered and used against hundreds of production systems before the information about its existence becomes public, much before a workaround or patch to avoid this threat is developed and this solution is implemented in a production system. That is why we should be prepared against this threat, above all, if it is a must for our business to be accessible from Internet. If the latter is the case, it would be desirable a greater control level on the security mechanisms that are established in our host.

To this aim, several security kernel modifications have been developed along the time and I will describe in the following sections. First, I will explore several compilation options that the Linux kernel offers to enhance system security. Secondly, I will review the most important kernel modifications sorted according to three trends.

Secure kernel compilation

The Linux kernel is a piece of software highly customizable. Therefore, it seems to be a good starting point to use the most secure possible kernel setup before enforcing any additional security mechanism. There are many tutorials and HOWTO's across the Internet that analyze significant kernel parameters and suggest values they should take to avoid security risks. These parameters usually are about customizing the TCP/IP stack and some kernel devices[17].

At this point, I would like to remind that the philosophy used for building a bastion host could also be applied to kernel hardening. In order to follow the

© SANS Institute 2003. As part of GIAC practical repository. Author retains full rights.
© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

principle of “Keep it simple” system administrators should remove from the kernel all pieces of software that are not necessary to the normal running of the system. That includes ATM drivers if only Ethernet networking is used or infrared and sound drivers for hosts that are physically located in a datacenter.

Several actions could be taken at this stage to enforce the second principle “Be prepared for the bastion host to be compromised”. It is always a good idea to enable system auditing because it might be useful both to detect suspicious user behavior and to keep track of potential intruder activities. In addition, system administrators should compile all necessary drivers within the kernel and disable the dynamic module loader whenever possible. An intruder will be unable to load most of rootkit kernel modules, as Knark[6] or heroin.c[25], to hide his presence on systems using a kernel configured in this manner.

Security feature-addendum modifications

This category of modifications offers the possibility of adding several new possibilities to the standard Linux kernel. The simplest example is the Openwall patch[23]. This patch offers additional security features to the standard Linux kernel. Using this patch, system administrators can decide to mark processes’ stack area as non-executable in order to to prevent most buffer overflows application bugs, restrict hard links at tmp directory in order to make unauthorized file access in this directory harder, or limit the use of FIFO’s at tmp directory to avoid FIFO write spoof attacks. This patch does not use any configuration file because it is fully configured at kernel compiling time. After installing and configuring this patch any further additional setup is needed when new software is installed. However, system administrators should make sure that security features that have been enabled do not break existing or potential applications.

There are several Linux distributions based on this patch. OpenWall Linux[22], Owl, is the most important of them and makes use of this patch by default. In addition, the source code of several components has been improved to match basic security standards. These components include SUID or SGID binaries, relevant code in system libraries, daemons and net services. As last improvement, this distribution brings into play a basic Trusted Computer Base, TCB, as alternative to classic UNIX password storage.

Linux Intrusion Detection System, LIDS is another patch that not only provides a larger amount of enhancements to standard Linux kernel but also offers mechanisms to impose some additional restrictions to subject-to-object relationships like Mandatory Access Control, MAC, and capabilities.

MAC is an access control mechanism additional to classic UNIX permissions, also known as DAC or Discretionary Access Control. Using MAC, system administrators can protect critical files against any kind of access. Moreover, they could even simulate the inexistence of those files. Unfortunately, this smart feature cannot be applied to the whole system because surely some applications need some access to these protected files. So, system

© SANS Institute 2003. As part of GIAC practical repository. Author retains full rights.
© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

administrators should identify these applications and configure LIDS to grant them appropriate access permission[11].

Unlike classical UNIX systems where root user has all the privileges, LIDS provides a fine-grained privilege separation called capabilities[12]. System administrators can assign certain capabilities to applications in order to enable them to perform limited privileged tasks that normally only the root user is allowed to accomplish. For example, the CAP_BIND_NET_SERVICE capability, when granted to a process, will allow it to be bound to a privileged socket (<1024). In this manner web servers, which need to be bound to port 80, could be executed by regular users without requiring any further privilege.

Additionally, LIDS provides a very useful feature called "kernel sealing"[10]. System administrators can configure LIDS to avoid any additional kernel module from being loaded after the kernel is initialized. Using this feature, the use of rootkit modules is highly prevented.

Using different patches at the same system is not incompatible as long as they affect separate areas of the kernel. In fact, there is a Linux distribution, EnGarde Secure Linux[9], which implements both the Openwall patch and LIDS.

Compartment modifications

Even if all the good-practices have been taken into account, it is possible that a host be compromised because, as we stated in previous sections, applications are not guarantied to be free from security flaws. So, system administrators must not forget the second principle of building a bastion host and should be ready for these situations.

A methodology commonly used to enforce this principle is "Compartment". It follows a simple philosophy that comes from comparing a host to a submarine. If one of the compartments of a submarine has a leak it is yet possible to seal it before the entire craft is compromised. In the same way, if a software subsystem suffers an attack, it would be desirable that intruders are lock up in a compartment designed for that subsystem to avoid they get full control of the host. Classic UNIX's do not follow this philosophy, e.g. if an intruder exploits a bug in a SIUD application and gets root access then the entire host is under his control.

A kernel extension designed to provide compartment is SubDomain. Using SubDomain[7] we can define a box, known as domain, for every non-trusted application we wish to jail. For every domain there is a configuration file where files and type of access are defined not only for the process itself but also for its child processes in a very easy way. SubDomain modifies the relevant system calls to perform further inspection before access is granted. This additional checking is always additional to the kernel native access control, so both access controls must be passed before a process is allow access to a resource. SubDomain is really light compared with other compartment products so it both follows the principle of "keep it simple" and does not have a severe impact in performance. There is a Linux distribution, Immunix OS[32], which includes the SubDomain patch preinstalled in the kernel.

© SANS Institute 2003. As part of GIAC practical repository. Author retains full rights.
© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Argus PitBull LX[1] is a commercial product for Linux. It includes a modified kernel that can be customized to fit your requirements and a set of utilities to configure several security options. Besides implementing a MAC access type, the system administrator can define several compartments or virtual machines named, as in SubDomain, “domains”. Within a domain, system administrators can confine, files, users, but also network resources and interfaces that can be completely isolated from other compartments. This is the main difference with previous products. Using PitBull, it is possible to group all files needed to run bind, port 53 and NIC 0 in a domain. In that way, if bind is violated, the intruder will be unable to access any other information like /etc/passwd file. It is also possible to set controls based on Interprocess Communications, IPC, allowing only authorized communications between process using pipes, messages queues, … These features offer an additional flexibility to contain potential intruders, however, they are difficult to configure and implement.

General frameworks

The security enhanced access mechanisms previously described have been created and developed independently, following different philosophies and focusing on particular problems. However, Linux, as general-purpose operating system, must satisfy a wide rage of user requirements and should be able to permit several access control models. Any of the previous kernel modifications addressed the requirement of creating a standard framework in which all the existing security improvements and the coming ones could be integrated. Maybe, that is the reason for which any of them have achieved acceptance within the security community.

In 2001, the National Security Agency presented a research prototype called SE Linux[21]. SE Linux is a group of kernel patches and modified utilities that can be used to control an extremely granular set of security rules. In this way, SE Linux provides role-based MAC, type enforcement and compartment, also know as multilevel security. This kernel prototype was based in the Flask architecture, the result of several OS security research projects undertaken by the NSA, Secure Computing Corporation and the University of Utah. They developed SE Linux in an attempt to transfer this technology to a larger developer and user community. But, although the philosophy established by SE Linux was very flexible, this first attempt, as previous works, was centered in the mere modification of the kernel to obtain some specific results. This first prototype was not designed as a proposal to modify Linux kernel in a way that any person that develops new improvements or security models could integrate them without interfering with other models developed by others together and in a simple way.

With this objective in mind, Linus Torvalds after SE Linux presentation, proposed several modifications to the Linux kernel structures in order to allow the implementation of SE Linux-like security models in the kernel development mainstream. That was the origin of the Linux Security Modules project, LSM. The main objective of this project is to develop a general-purpose security access control framework for Linux kernels, which would allow system administrators to load and execute several control models. The

© SANS Institute 2003. As part of GIAC practical repository. Author retains full rights.
© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

solution proposed is really general and its performance impact is minimum. The following were the desirable features defined for this framework

- This framework should be generic enough to support several security models.
- It should be light in terms of performance.
- It must support access control models.
- It is necessary to support previous functionalities.

These modifications consisted of adding a set of fields to the structures of main kernel objects, as process, file systems, pipes, files, sockets and IPC mechanisms, in order to store their security attributes. In addition, a new function pointer array was added to the kernel both to enable read-write accesses over these new attributes and to control actions that different system subjects are authorized to carry out. This array, by default, contains traditional functions utilized in the operating system, so the default behavior is identical to the classical Unix. When a kernel security module is loaded a new kernel function is in charge of modifying this function array so that its entries point now to the new loaded code.

The first loaded module in a system is called primary security module. This module is in charge of managing basic security functions but, in the framework proposed by Torvalds, should also implement the necessary logic so that additional security modules, that potentially might be loaded, are capable of cooperating among them. In this way, if the primary security module allows it, it is possible to load a second module that implements an advanced policy on process management and even a third one that runs a different model to access IPC mechanism. At present, several kernel modifications that we have mentioned previously, as SE Linux, LIDS or Openwall, have already been ported to security modules following the rules defined by the LSM project[31]. As a result, system administrators have already available several security models to choose from.

Using LSM, anything a system administrator could imagine is possible. He can restrict network access, limit file access, allow root user to only perform certain restricted tasks… On the other hand, this software is neither trivial nor easy to use. Deep knowledge about security matters and application internals is a must in order to make a careful system security planning. Besides, this general framework only focuses on control access issues. A security policy must include many other technical subjects, as auditing, that are no covered yet by the LSM.

© SANS Institute 2003. As part of GIAC practical repository. Author retains full rights.
© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Conclusions

Through this paper we have reviewed several tools and resources that system administrators could use to improve standard security access in Linux. But even if you enable most of the previously described security mechanisms it does not mean that your host is highly secure. Most of the times quality is more important than quantity. It is completely worthless to install all previous enhancements if they are not correctly configured to address your security policy or even worse if that security policy does not exist. The right approach is, first, to define your security policy and then to choose the most suitable mechanism to implement it.

As a last recommendation, keep in mind that the target of every hardening process is to get a suitable balance between a certain security level and administration facilities. Therefore, system administrators must decide, at each step of any hardening process, how far they are willing to go. Everybody seems to agree that security is important, but even more important is that our business application keeps its functionality. If you do not know how to reach this balance, it is advised to go step by step. After complexion of each step, you should verify that all the required business functionalities of your application are still available before moving on to the next stage.

© SANS Institute 2003, Author retains full rights.

© SANS Institute 2003. As part of GIAC practical repository. Author retains full rights.
© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

References

1. Argus Systems Group. “Products White Papers PitBull LX”. May 9, 2003. URL: http://www.argus-systems.com/product/white_paper/lx (Jan 15, 2003)
2. Astaro AG. “Astaro Security Linux”. URL: http://www.astaro.com/php/statics.php?action=asl&lang=gb (Feb 18, 2003)
3. Baratloo, Arash; and others. “Libsafe: Protecting Critical Elements of Stacks”. URL: http://www.research.avayalabs.com/project/libsafe/doc/libsafe.pdf (Mar 7, 2003)
4. Center for Internet Security, The. Linux Benchmark v1.0.0. Security Essentials Day 6 – SANS Track One. The SANS Institute. 2002
5. Chapman, Brent and Zwicky, Elizabeth. Building Internet Firewalls. O’Really Associates. First Edition, November 1995
6. Clemens, Jonathan. “Knark: Linux Kernel Subversion”. URL: http://www.sans.org/resources/idfaq/knark.php (Mar 5, 2003)
7. Cowan, Crispin; and others. “SubDomain: Parsimonious Server Security”. URL: http://www.immunix.org/subdomain.pdf (Jan 19, 2003)
8. Frazier, Frank. “Frazierwall Linux 3.5a”. URL: http://www.frazierwall.com (Jan 18, 2003)
9. Guardian Digital, Inc. “EnGarde Secure Linux: Features and Benefits”. URL: http://store.guardiandigital.com/html/eng/products/software/esp_features.shtml (Feb 18, 2003)
10. Hatch, Brian. “An Overview of LIDS, Part One”. Oct 17, 2001. URL: http://www.securityfocus.com/infocus/1496 (Mar 5, 2003)
11. Hatch, Brian. “An Overview of LIDS, Part Two”. Oct 31, 2001. URL: http://www.securityfocus.com/infocus/1502 (Mar 5, 2003)
12. Hatch, Brian. “An Overview of LIDS, Part Three”. Nov 12, 2001. URL: http://www.securityfocus.com/infocus/1510 (Mar 5, 2003)
13. Hatch, Brian. “An Overview of LIDS, Part Four”. Nov 29, 2001. URL: http://www.securityfocus.com/infocus/1517 (Mar 5, 2003)
14. Huagang, Xie. “Build a Secure System with LIDS”. Oct 4, 2000. URL: http://www.lids.org/document/build_lids-0.2.html (Mar 1, 2003)
15. Lasser, Jon. “Bastille Linux”. URL: http://www.bastille-linux.org .(Dec 17, 2002)
16. Linux-Consulting. “Hardening and Tightening Security on Your Server/Network”. URL: http://www.linux-sec.net/Harden/harden.gwif.html (Dec 17, 2002)
17. Linux Documentation Project. “Kernel Security”. Linux Security HOWTO. URL: http://www.linux.org/docs/ldp/howto/Security-HOWTO/kernel-security.html (Feb 20, 2003)
18. Lübbert, Jörg. “Information about Kaladix Linux”. URL: http://www.kaladix.org/docs/information.shtml (Feb 10,2003)
19. Mandrake Soft. "Mandrake Linux - Friendly Linux operating system for both servers and desktop". URL: http://www.mandrakelinux.com (Mar 30, 2003)
20. Meyer, Frank. “What is Fli4L”. URL: http://www.fli4l.de/english/e_fli4l.htm (30 Jan, 2003)
21. National Security Agency, The. “Security-Enhanced Linux”. URL: http://www.nsa.gov/selinux/index.html (Jan 20, 2003)
22. Openwall Project. “Owl, -- a security-enhanced server platform”. URL: http://www.openwall.com/Owl/CONCEPTS.shtml (Feb 27, 2003)

© SANS Institute 2003. As part of GIAC practical repository. Author retains full rights.
© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

23. Openwall Project. “Openwall GNU/*/Linux presentation slides” URL: http://www.openwall.com/presentations/Owl/ (Feb 27, 2003)

24. Red Hat, Inc. "Red Hat -- Linux, Embedded Linux and Open Source Solutions". URL: http://www.redhat.com (Mar 30, 2003)

25. Samhain Labs. “Loadable Kernel Module (LKM) Rootkits”. URL: http://la-samhna.de/library/lkm.html (Mar 1, 2003)

26. Smalley, Stephen; and others. “Linux Security Modules: General Security Hooks for Linux”. URL: http://lsm.immunix.org/docs/overview/linuxsecuritymodule.html (Jan 20, 2003)

27. SuSE Inc. "SuSE, The Linux Experts". URL: http://www.suse.com/ (Mar 30, 2003)

28. Trustix AS. “Trustix Secure Linux 1.5 Users Guide”. URL: http://www.trustix.net/pub/Trustix/current/i586/doc/html/tsl-doc/tsl-doc.html (Feb 15, 2003)

29. Wing, Chris. “New features in CAEN Linux 6.1”. URL: http://www.engin.umich.edu/caen/systems/Linux/caenlinux/6.1/features.html (Feb 10, 2003)

30. Wiresoft. “Firegate Server SMB Edition”. URL: http://www.wiresoft.net/Firegate_White_Paper.PDF (Feb 15, 2003)

31. WireX Communications Inc. “Current List of Modules Ported to LSM”. Linux Security Modules URL: http://lsm.immunix.org/lsm_modules.html (Jan 20, 2003)

32. WireX Communications Inc. “Immunix Secured Linux 7+” URL: http://www.immunix.org/immunix7+.html (Apr 5,2003)

33. WireX Communications Inc. “Immunix Security Technology”. URL: http://www.wirex.com/Immunix/index.html (Feb 18, 2003)

34. WireX Communications Inc. “StackGuard Mechanism: Stack Integrity Checking”. URL: http://www.immunix.org/StackGuard/mechanism.html (Mar 5, 2003)

35. Zelow Consulting. “floppyfw”. Feb 26, 2003. URL: http://www.zelow.no/floppyfw (Feb 10, 2003)

36. Zuerker, Heiko; and others. “Devil-Linux Documentation”. Aug 30, 2002. URL: http://www.devil-linux.org/ADMIN.html (Feb 3, 2003)

© SANS Institute 2003. As part of GIAC practical repository. Author retains full rights.
© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Last Updated: September 2nd, 2015

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Crystal City 2015	Crystal City, VAUS	Sep 08, 2015 - Sep 13, 2015	Live Event
SANS Network Security 2015	Las Vegas, NVUS	Sep 12, 2015 - Sep 21, 2015	Live Event
SANS Seoul 2015	Seoul, KR	Sep 14, 2015 - Sep 19, 2015	Live Event
Data Breach Investigation Summit & Training	Dallas, TXUS	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Baltimore 2015	Baltimore, MDUS	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Perth 2015	Perth, AU	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Tallinn 2015	Tallinn, EE	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS ICS Amsterdam 2015	Amsterdam, NL	Sep 22, 2015 - Sep 28, 2015	Live Event
SANS Bangalore 2015	Bangalore, IN	Sep 28, 2015 - Oct 17, 2015	Live Event
SANS DFIR Prague 2015	Prague, CZ	Oct 05, 2015 - Oct 17, 2015	Live Event
SANS Seattle 2015	Seattle, WAUS	Oct 05, 2015 - Oct 10, 2015	Live Event
SANS Tysons Corner 2015	Tysons Corner, VAUS	Oct 12, 2015 - Oct 17, 2015	Live Event
SOS: SANS October Singapore 2015	Singapore, SG	Oct 12, 2015 - Oct 24, 2015	Live Event
SANS Gulf Region 2015	Dubai, AE	Oct 17, 2015 - Oct 29, 2015	Live Event
SANS Cyber Defense San Diego 2015	San Diego, CAUS	Oct 19, 2015 - Oct 24, 2015	Live Event
SANS Tokyo Autumn 2015	Tokyo, JP	Oct 19, 2015 - Oct 31, 2015	Live Event
SANS South Florida 2015	Fort Lauderdale, FLUS	Nov 09, 2015 - Nov 14, 2015	Live Event
SANS Sydney 2015	Sydney, AU	Nov 09, 2015 - Nov 21, 2015	Live Event
SANS London 2015	London, GB	Nov 14, 2015 - Nov 23, 2015	Live Event
Pen Test Hackfest Summit & Training	Alexandria, VAUS	Nov 16, 2015 - Nov 23, 2015	Live Event
SANS Hyderabad 2015	Hyderabad, IN	Nov 24, 2015 - Dec 04, 2015	Live Event
SANS Cape Town 2015	Cape Town, ZA	Nov 30, 2015 - Dec 05, 2015	Live Event
SANS San Francisco 2015	San Francisco, CAUS	Nov 30, 2015 - Dec 05, 2015	Live Event
SANS Milan 2015	OnlineIT	Sep 07, 2015 - Sep 12, 2015	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced