

Linux Security for the Enterprise: Executive Summary

Trent Jaeger, David Safford, Hubertus Franke
IBM T.J. Watson Research Center
Yorktown Heights, NY
{jaegert,safford,franke}@us.ibm.com

Summary

This white paper summarizes the currently available security features in typical Linux distributions, in addition to other security services and products that can be added. We also discuss ongoing research and development in future versions of Linux that will address emerging security requirements. In this summary, we focus on security requirements relevant to enterprise-level Internet servers. We show that current Linux distributions provide good security features, equivalent to other comparable operating systems. In addition, numerous optional services and products have been developed that provide further enterprise-level security functionality. The Linux community has also recognized the need for a single framework upon which this much of this functionality can be leveraged, called the Linux Security Modules (LSM). IBM should actively participate in making LSM a viable framework for meeting enterprise requirements and help the Linux community add further services to Linux to enable future systems to substantially reduce e-business security risks.

The Security Problem

Modern e-commerce servers have to provide strong guarantees of both the secrecy of data, (for sensitive data like credit card numbers), and the integrity of data (for things like web pages). Unfortunately, hackers have been increasingly targeting these servers, finding ways to break through the secrecy and integrity protection. In recent analysis, CERT found that hackers have been finding new vulnerabilities at dramatically increasing rates[], and that they have been exploiting these problems at similarly alarming rates[]. The FBI reports that credit card theft from web servers has become the fastest growing source of financial fraud.

Security Features in Current Linux Distributions

Linux has a variety of security features that provide comparable security to other existing operating systems. Linux contains basic security features, such as password authentication, file system discretionary access control, and security auditing. These three features are necessary to achieve a security evaluation at the C2 level. Most commercial server-level operating systems, including AIX, Windows NT, and Solaris, have been certified to this C2 level. While Linux has a few minor omissions in its auditing and password management features, it is largely C2 compliant, and could be certified at this level with only minor tweaks.

In addition, new security features, such as automated software updating, packet filtering, and more secure configuration definitions, have been added for recent versions of Linux to start to address most common security problems. This section describes some of the basic security

features of Linux (see Romans and Ratliff[7] and Chapman[3] for good summaries of Linux security features from which most of this section is derived). Many additional products and extensions to Linux are under development which will further improve its security, but services that are not yet part of a typical Linux distribution are described in the next section.

First, a significant infrastructure exists for the dissemination of software bug reports and patches. Further, distributors are providing tools to help distribute patches automatically (e.g., RedHat's **UP2DATE** tool). Second, improved firewall options are now available in Linux distributions, such as using the tcpwrappers daemon to protect services called via *inetd* and the NetFilter stateful firewall (derived from OpenBSD's iptables). The former technology enables verification, authorization, and logging of remote requests. The latter technology is a more comprehensive firewall system that is also able to block some types of denial-of-service attacks. Thirdly, unlike many commercial operating systems, Linux has a state-of-the-art kernel source of true random numbers, which is critical for good quality cryptographic services.

Security Threats

While Linux is comparable to other server operating systems in its security features, it can be made more resistant to current hacking threats. To understand how additional features can help, one needs to understand the nature of the current threats. Server systems normally become vulnerable in one of two ways: through buggy software or through misconfiguration.

By far, the most common attack on server machines is to exploit known bugs or vulnerabilities in the software providing one of the services. Network services inherently accept request data over the network, even from potentially malicious sources. If the software is not careful with this malicious data, the data may be able to exploit the lack of care to "take over" the server process, making it run code of the hacker's choosing. Two common categories of such "data driven" attacks are buffer overflows and parsing errors. While the existence of these has been known for nearly 30 years and the fixes are well-understood, these kinds of bugs are continuously found in software.

Misconfiguration is the second major way in which servers can become vulnerable to attack. In misconfiguration, a service is not buggy, but is accidentally configured to give out services that it shouldn't. For example, a web server normally will not allow a client to access sensitive data like the password file, but the server may accidentally be misconfigured to allow this access. While it may seem simple to configure a service correctly, in practice this can be quite difficult.

The first challenge is that there is no standard way to express a desired security policy across all services on a server. Each service tends to have its own arcane configuration methods. On a typical Linux server, there are dozens of security critical configuration files spread throughout the filesystem, which all must be set correctly. If any one of them has an error, the entire system can be vulnerable.

Even if an administrator manages to configure a machine securely, it is very easy to cause subsequent inadvertent misconfiguration. Every installation of even minor new applications or

application updates can change or overwrite configuration files. Major applications have been known to install security nightmares, such as setuid root executables and world writable executables. This configuration problem is made almost impossible by the need to install all too frequent security critical patches.

Inadequacies in the current security features

While some people claim that the existing security features in Linux and other available operating systems are adequate to meet the demands of e-commerce security, there is clearly room for improvement. One popular argument is that servers are vulnerable only because lazy administrators fail to keep up to date with the latest security patches. The problem with this argument is that patches come out too frequently - seven per day, across all systems. Thus, an administrator is often faced with the choice of applying patches without adequate testing, and possibly breaking a critical application, or not applying the patch, and being hacked.

A second common security misconception is to place too much trust in security tools, such as firewalls and intrusion detection systems (IDS). A firewall does protect against some attacks, but many other attacks, such as e-mail and web attacks go right through. While an IDS can detect many intrusion attempts, it often notices the attack only after it has succeeded. This is little comfort after customer credit card numbers have been stolen.

Administrators need additional tools, beyond the standard operating systems, to manage the security of their systems. These tools must provide the administrators with easily managed security configurations, provide a coherent approach to software update, contain the effects of buggy software, and enable effective use of auditing and intrusion detection.

Linux Security Requirements

A Linux that is secure against these threats must:

- Contain no vulnerabilities
- Authenticate all users and programs
- Authorize all security-sensitive kernel operations
- Protect the trusted computing base (TCB) from modification

If Linux itself contains vulnerabilities, then attackers can exploit these to subvert any security policy. While this is very difficult to prove in practice, measures should be taken to minimize vulnerabilities. The fact that Linux is open source software will be beneficial in this case. It is necessary to authenticate all users, be they remote or local, and the programs that they use in order to make decisions correctly about what permissions they should be granted. Each security-sensitive operation must be authorized in order to ensure that such operations are only executed by subjects with the required permissions. The Linux kernel, its authentication and authorization systems, and any other programs and data used in security decisions constitute the *trusted computing base* (TCB) of the system. We must ensure that the Linux TCB can protect itself from unauthorized modification.

The behavior of the authentication and authorization systems depend on the system's configuration, in particular, its security policies. An environment that enables the design of a secure Linux configuration must have:

- Simple policy selection at installation
- Easy to understand policy models and choices (or the policies must be immutable)
- Use policies that cannot be subverted by users or application installation

Linux should be installed in a configuration that is known to be secure for the target application domain. The policy model must either be immutable, simple enough to understand, or provide tools sufficient to verify easily understood security properties. Lastly, the security policies themselves must provide protection from their subversion in the same way as the TCB, of which they are ultimately part. A policy which can only be modified by system administrators is referred to as a *mandatory access control* (MAC) policy. Thus, support for MAC policies is required of a secure Linux system.

Finally, tools such as auditing and IDS can provide assistance by detecting and acting when authorized subjects (perhaps due to compromise) try to use their special privileges in unusual ways. For example, any operations that could result in the creation of a system trapdoor must be reliably audited.

Detailed Linux security requirements are described in the accompanying technical report.

Linux Security Extensions

Historically, commercial vendors have implemented “trusted” versions of their operating systems that were targeted at government, particularly defense, agencies. The main modifications to these systems were the addition of a particular kind of mandatory access control model, called a *multi-level security* (MLS) model, and auditing capabilities. A MLS model is designed to prevent the leakage of data to unauthorized subjects, but does not address the integrity of the system. That is, such systems prevent the leakage of data (e.g., credit cards), but do not prevent the exploitation of bugs by user data from untrusted sources that may compromise the entire system which is also a requirement of enterprise systems. Therefore, these systems not applicable as is to enterprise requirements.

A number of advances have been implemented that start to address enterprise requirements. First, the **TCPA** consortium[9] aims to provide hardware support for reliable system integrity verification, and tamperproof platforms, such as **IBM's 4758**[5], enable the installation of security services in hostile environments. A large number of advances have been made for Linux as well. **Bastille Linux**[2] and **Immunix**[6] provide hardening tools for preventing bugs from taking over system services that run as root. For example, Immunix is a family of tools designed to cause system services to fail safely when one of a variety of common vulnerability types (e.g., buffer overflow) is used in an attack on them. **Trustix**[10] and **HP Secure OS Software for Linux**[4] provide a variety of tools to improve Linux security (e.g., TripWire and encrypted file

systems). Many of these systems provide MAC policy models for files, but only **Argus PitBull**[1] provides a model that enables control of network objects as well. The MAC policy models used in these systems are either significantly limited or complex and lack supporting management tools. Trustix does provide support for assisting system administrators in installing a system with minimal services and preventing accidental initiation of new services. Argus PitBull is one of the few systems that leverages the Pluggable Authentication Modules (PAM) framework in Linux explicitly to derive its subject identities. However, PAM does not itself have an authenticated channel to the kernel (although it is the best thing available now). Also, no system provides support for verifying the integrity of the programs that they use.

A fundamental problem with all of the approaches above is that they require kernel modifications to provide the desired authorization flexibility and performance. This is because the actual objects and operations performed are determined deep inside the kernel, so the kernel must be changed to ensure that the intended policy is enforced. Because each system uses different, ad hoc kernel modifications none will be accepted into the base kernel.

The Future of Linux Security

Work is underway to provide an acceptable, base kernel authorization framework behind which a variety of policies can be supported. The Linux Security Modules (LSM) framework adds authorization hooks into the base Linux kernel that intend to cover every controlled operation in Linux kernel. The hooks are independent of the authorization policy, so a variety of MAC policies can be supported.

LSM is not yet an accepted part of the base Linux kernel, although Linus has expressed willingness to accept authorization hooks that are implemented as generic function pointers, as LSM is. The LSM project is trying to gain acceptance for the Linux 2.5 kernel. Policy modules are currently under development for Immunix's SubDomain policy and SELinux's DTE policy[8].

Improving Linux Security Based on LSM

Fundamentally, LSM is only an authorization framework, so many other features are necessary to build a secure Linux system. First, we identify the tasks that can be performed to install an initial TCB for a secure Linux system. We indicate where Linux extension systems provide similar functionality.

- Verify that the LSM hooks implement the necessary security checks in the Linux kernel
- Run hardening tools over Linux system services (e.g., Bastille Linux and Immunix)
- Identify trustworthy and sufficient LSM modules
- Integrate commensurate authentication functionality in the kernel with LSM
- Select an initial policy configuration based on application type
- Select an initial system installation that initiates only minimal services (e.g., Trustix)

Next, we list a set of tasks that should be undertaken to maintain the integrity of the TCB.

- Require administrative authorization over the initiation of system services (e.g., Trustix)
- Verify the integrity of the TCB and TCB data
- Provide policy management tools to verify policy properties, such as safety
- Audit and perform intrusion detection on all operations on TCB data

Finally, we outline the use of the secure Linux system.

- Boot the system securely based on the risks of the environment (e.g., TCPA or tamperproof hardware in environment where hardware attacks are possible)
- Authenticate users via authentication module and assign initial subject identifier
- Verify the integrity of programs run by the user and re-assign subject identifiers for the combination of user and program
- Verify the source and integrity of data used by these programs and determine whether the subject identifier should be changed
- Authorize all controlled operations from LSM hooks based on the subject and object identifiers in LSM module
- Audit all administrative operations and provide the ability to audit all controlled operations

Linux itself should be shown to be sufficient to implement necessary authentications, authorizations, and integrity checks. Then, policies that implement a basic configuration can be selected. This configuration should have simple semantics and initiate minimal functionality. Changes to the configuration must be controlled and evaluated to enable system administrators to maintain system security. Finally, use of the Linux system involves a complete and coherent policy enforcement that can take into account users, the programs they execute, and the data that they use.

A detailed of this Linux security architecture is contained in the accompanying technical report.

Towards Secure Linux for the Enterprise

The LSM community is growing and making progress toward their goal of getting LSM into the Linux base kernel. The LSM community is led by Wirex with DARPA sponsorship. NAI Labs with NSA sponsorship has been another major contributor, but many other companies and individuals from around the world have also provided input to the LSM framework. IBM has a unique opportunity to help the LSM community develop a framework upon which enterprise security will be based. Ongoing relevant IBM work includes tools for verification of LSM hook placement, program integrity verification, and scalability analysis.

Once LSM gains acceptance, a complete security solution for enterprise Linux can be developed. IBM should also contribute its experience in tamperproof hardware, secure booting, intrusion detection, and policy management. We envision that the development and management of security policies for LSM will be a key area of work. LSM gives the community a powerful basis for managing system resources, but system configuration must be improved to reduce the degree of misconfiguration currently seen.

References:

1. Argus PitBull. <http://www.argus-systems.com/product/>.
2. Bastille Linux. <http://www.bastille-linux.org>.
3. Chapman. Addressing security issues in Linux. IBM whitepaper. <http://www.ibm.com/developerworks/oss/whitepapers/Linux-Security-IBM-White-Paper.pdf>.
4. HP Secure OS Software for Linux. <http://www.hp.com/security/products/linux/>.
5. IBM 4758. <http://oss.software.ibm.com/developerworks/opensource/4758/>.
6. Immunix. <http://www.wirex.com>.
7. Robb Romans and Emily Ratliff. Linux Security “State of the Union.” IBM whitepaper. <http://www.ibm.com/developerworks/oss/whitepapers/LTC-Security-Whitepaper-external.pdf>.
8. SELinux. <http://www.nsa.gov/selinux/>.
9. T CPA. <http://www.trustedpc.org>.
10. Trustix. <http://www.trustix.com>.