Global Information Assurance Certification Paper

GIAC Certified UNIX Security Administrator (GCUX)
Practical Assignment
Version 2.0, Option 3

The not so phantom menace

Improving UNIX security from a Windows Perspective

Jorge D. Ortiz-Fuentes

May 2, 2004

Abstract

UNIX administrators live in a very UNIX-centric world, but there is a world outside and there are a lot of Windows systems out in this world.

Microsoft has invested a considerable amount of money in improving Windows security. Some of the features they have added were already present in most UNIX systems, but some others are worth a review.

For every security feature that I include in my wish list, I will explain the concept behind it, make up an example where it is useful, present the currently available alternatives for Linux systems if any, and define what is still missing.

Contents

1 Introduction

Let's face it! Most UNIX administrators feel strongly that their systems are so superior to their Windows counterparts that they won't even consider they have security features that are great and concepts that should become part of regular UNIX systems. Besides missing the opportunity to be more valuable for their employers trying to be able to handle problems in both systems, they also fail to identify good security features that are not implemented in their proudly administered UNIX systems.

Being a UNIX person myself, I think that we should look at the security features offered by Windows to learn from their experience and improve our UNIX systems.

Throughout this paper I will focus on Linux as a UNIX-like operating system to see if the —Windows— security features that I mention are available for it. In particular I will talk about the Fedora Core 1 Linux distribution, because it is available for free and is fairly recent. Some of the things will apply also to other Linux distributions as well as some other UNIX operating systems (HP-UX, Solaris, AIX, FreeBSD. . .)

For every security feature that I include in my wish list, I will explain the concept behind it, make up an example where it is useful, present the currently available alternatives for Linux systems if any, and define what is still missing.

1.1 Acknowledgments

The unquestionable help that Rosa and Lidia have provided me with, through their patience and unconditional support, has been the decisive factor for writing this paper.

I also thank David and Raul. Working together as a group is making us all improve and extend our knowledge about security.

Finally, I sincerely thank all the people that have put their knowledge and work in developing the free tools that I use every day and that are both an important part of what I explain in this paper and the applications used for writing it.

2 The wish list

2.1 Access Control Lists (ACLs)

Here I will explain how ACLs can help over regular UNIX permissions, which Linux file systems include —or are able to include— ACL support[1], and what is not ready yet.

2.1.1 Concept and usage

UNIX is built around the paradigm that everything is represented as a file. Not only regular files, programs or links, but also directories, named pipes or FIFOs, sockets, and even devices. Some UNIX variants, even offer a file system representation

of the kernel structures, implemented in Linux as the `/proc` file system. Hence the interest of the operating system designers and users to control access to each file.

The traditional security mechanism used in UNIX to allow or deny access to a file or directory is the use of permissions. Together with the data that the file contains, other information is stored, like the file name or the last time it was accessed. This additional information is the metadata, the data that describes the real data. UNIX permissions are a mandatory part of the metadata of each file.

Each user is identified by an integer number called user id (UID). Similarly, groups are identified by an integer called group id (GID). Each file **must** belong to a user and a group and three regular permissions are defined with respect to the user, the group and everybody else. Every process runs associated to a UID and one or more GIDs, and when it tries to access a file, if the process UID and the file UID are equal the user permissions define the access. If the UIDs were different but any of the GIDs is equal to the one owning the file the access is controlled by the permissions granted to the owner group. Otherwise the permissions granted for the others rule.

The regular permissions allow or deny access to read, write and execute execute the file or transverse the directory. Three additional permissions can be set to cover specific functionalities: the setuid, the setgid and the sticky bits.

This mechanism has multiple advantages:

- It is fairly simple to understand and be applied.
- It is easy to represent the permissions in a compact way, which makes auditing and verifying them much easier.
- Implementation is less complicated and since the metadata has a fixed size, the checks can be implemented in a very efficient and clear way.

However, it also has disadvantages:

- The semantics are quite restrictive: it only allows to define the permissions for one group.
- The granularity level is low: it only allows to define permissions to read, write and execute.

Windows solves this problem implementing access control through *Discretionary Access Control Lists* (DACLs). The DACL is composed of *Access Control Entries* (ACEs). Each ACE defines an allowed or denied access for a user or, preferably, a group. Denied accesses take precedence. If a user belongs to two groups and one of them is allowed to read a file and the other one is explicitly denied to do so, she cannot read the file.

Access control lists are a useful feature for an operating system that provides simple solutions to complex access control scenarios. The simplest of these scenarios could involve two groups. One of them should be allowed to create and modify documentation. The other group should only be able to read it. Nobody else should read this documentation. Regular UNIX permissions do not provide a straight solution to this problem.

2.1.2 Existing Linux solutions

ACLs must be controlled at the kernel level, as regular UNIX permissions are. Every time a file needs to be accessed the kernel takes care of the process through a system call. During the execution of the system call the permissions are checked and enforced.

Linux implementation of ACLs is based in an unfinished version of a POSIX standard, particularly the drafts of the IEEE standards 1003.1e —security interfaces to open systems for access control lists among other things— and 1003.2c[2] —security utilities to open systems for access control lists among other things. While the work of those groups has been discontinued, most of it can still be used instead of reinventing the wheel to implement ACLs. Some other UNIX operating systems vendors like SGI (Irix) or Sun Microsystems (Solaris) seem to have the same opinion since they also based their implementations on the drafts.

The standard stable Linux kernel, the one that can be downloaded from the primary site http://kernel.org/, did not include this functionality until the recent 2.6 was released. And even the newest 2.6 version includes ACL support only for some file systems, like ext2 or ext3, but not for others, like NFS.

Nevertheless, there have been patches available to add this capability to previous versions of the kernel for a long time now. In particular, the previous stable release of the Linux kernel, that is the one in use in many Linux systems —only a few distributions have released their Linux versions using kernel 2.6 already,— can be modified to support ACLs by using the patches that Andreas Grünbacher has available for downloading[1]. He also has pointers to the required patches for other file systems to support ACLs.

2.1.3 The missing parts

Only a reduced subset of the Linux users patch their kernels by themselves, mainly to avoid non-reproducible problems in production environments. Most users prefer to delegate this task to their preferred distribution instead. However, up until now many Linux distributions did not include complete ACL support, if any.

Fedora Core 1 does not include the patches in the kernel. Fedora Core 2, that is going to be based on the 2.6 kernel, will include some support for ACLs.

Kernel support is an essential part of the ACL implementation, but other things are also necessary.

First, you need utilities to set and read the ACLs. The ones defined in the POSIX 1003.2c draft are `setfacl` and `getfacl` and are available for Fedora Core 1 in the acl package that comes with the distribution CDROMs.

The core utilities —basic commands that you expect installed in your Linux system like `ls` or `cp`— must be able to work with ACLs too. `ls` must be able to inform the user that a file has an ACL assigned and it does so with a plus (+) sign in the permissions field. `cp` must be able to copy the ACLs together with other metadata that belongs to the file.

The backup utilities must be able to save and restore the ACL information. There is a utility that is called `star` that is able to work with the ACL information[3].

`star` is already available in the distribution CDs of Fedora Core 1.

Finally, it is also very important to have file browsers, both graphical and text based, that support ACLs. Neither Nautilus nor Konqueror support ACLs. However, there is a project to offer graphical support for ACLs and that can be used from within Nautilus[4], and another one to extend Konqueror[5].

The current status of the other major Linux distributions can be found in the Table 1.

Distribution	Kernel	Core utils	File manager	Backup
Debian 3.0 (stable)	no	no	no	no
Debian 3.1 (testing)	yes (2.4 & 2.6)	yes	no	yes (star)
Red Hat RHEL 3.0	yes	yes	no	yes (star)
Suse 9.1	yes	yes	no	yes (star)

Table 1: Support for ACLs of each major Linux distribution

2.2 Audit capabilities for every object

In this section I will explain the advantages of being able to audit every kind of access to each object in the machine.

2.2.1 Concept and usage

It is an essential part of every system to be able to verify that things happen as you would expect, and to have alarms that go off when they do not. This is why audit is important: you do want to know what happens in your system.

Auditing has disadvantages too. Depending on the implementation, audit can generate an overwhelming amount of information. This affects both data storage and performance. The system needs dedicated disk space and bus bandwidth. CPU usage is also increased due to the extra amount of operations that are needed to take a decision on whether an audit event must be generated and, if so, do it.

On the one hand, mature UNIX systems like HP-UX implement audit at the system call level. A system call is a request to the kernel for one of the services that it offers, like opening a file or sending a signal to a process. The system call audit facility even offers the possibility to select which system calls are audited. So if you want to know when a file is accessed, you can audit the `sys_open` system call. This will, for sure, provide you with the information you want and much, too much more. You will get an event for each accessed file. Not only regular files, but also device files and libraries, for example.

The system call based audit conforms the C2 security level specification defined in the *Trusted Computer System Evaluation Criteria* (TCSEC). Among other things, C2 systems must be able to generate audit events for every:

- Use of the identification and authentication mechanisms.

- Modification the users address space, like opening a file or running a program.
- Deletion of an object, like deleting a file.
- Administrative action, like mounting a file system or shutting the system down.

But the most important part is that applications cannot make the decision on whether to generate an audit event or not. It must be mandatory, so it has to be the kernel the one that generates those events.

On the other hand, professional Windows operating systems target a different objective when using audit. Windows uses an object based audit approach. Most of the times object is a synonym of file, but it can also be a registry entry or an Active Directory object. Windows audit facility concentrates on file operations — other events should be generated independently of the auditing subsystem,— but it allows the administrator to select what you audit on a per-object basis instead of per system call.

You could get a similar behavior filtering the UNIX audit events before writing them to a file, but in this case:

1. All the events, with its corresponding performance impact, would have been generated
2. The filtering program would have to be very stable and able to handle a huge workload.

2.2.2 Existing Linux solutions

Fedora Core 1 does not include an audit facility like the ones described above. Fedora Core 2 includes the *Security Enhanced Linux* (SELinux) *Linux Security Module* (LSM) that is the result of an NSA project. Among the security features offered by the SELinux, it is included the ability to do system-call auditing. This is not exactly a C2 auditing capability, but a way to provide information about the system call that was being executed every time SELinux generates a denied message.

There was a open source project called *Secure Auditing for Linux* (SAL)[6] that tried to develop a kernel audit package that was compliant with the one required for a C2 system. However, no new files have been released for the last year and the last supported kernel is 2.4.18.

Recently, it has been a fair amount of activity in the Linux kernel list about this topic and even some patches for the current kernel have been submitted. The most relevant one is the lightweight auditing framework[7] that provides the mechanism for doing ruled based system call auditing. The rules indicate if an audit even should be generated based on:

- attributes of the process that is using the system call,
- return value of the system call,
- or even the parameters used in the system call, such as a file name.

2.2.3 The missing parts

On the one hand, system call audit is useful for artifact or forensic analysis, but provides too much information for permanent usage. On the other hand, object access audit is very useful too if you want to answer questions like who has accessed this file. Linux should have both available: the one that allows to audit system calls, which for simplicity I will call *audit-s*, and the one that allows to audit only selected file operations on a per-file basis, which I will call *audit-f*. A high level description of a possible implementation of each capability is detailed below.

Audit-s should be compiled with the kernel and enabled or disabled by software, probably via a /proc entry. When enabled, audit-s will keep an array in memory with an entry for each system call that will contain the logically or-ed flags of the operations to audit for the system call. The system call gate, which is the small piece of code that is used when a system call is done to select the piece of code to execute and to return back to user mode, should generate all the events.

Audit-f should also be compiled with the kernel and enabled or disabled by software. Each file that is to be audited should have stored with its metadata the operations to audit. This information can be stored using the extended attributes patch[1]. The operations to be audited are the same than the ones used for the UNIX permissions: read, write and execute. The audit events will also differentiate the user owning the file, the group owning the file and everybody else. Audit events could also be generated when the setuid, setgid and sticky bits are used. Each of the system calls involved in the operations that are to be audited (`sys_open`, `sys_exec`, etc.), will contain code that generates the corresponding audit events.

2.3 Administrative rights

In this section I will discuss the advantages of being able to do task separation instead of having an all-powerful administrative account.

2.3.1 Concept and usage

UNIX and Windows user models can be compared to the monotheism and polytheism respectively.

On one hand, in the UNIX world there are only two *levels of existence*. You can either be the almighty god that makes the rules or a simple mortal. When a mortal being needs anything special to happen, something that she cannot do by herself —like changing the rules of the universe or changing the permissions of somebody else's file— they pray and ask god to do it. Root, that is the name of god in the UNIX monotheistic religion, is the only one with the ability to do special things.

On the other hand, the Windows world has many more deities. The mortal beings are, like the ones in the UNIX world, unable to do special things. But every god has a only a small set of special powers. Some of them have only one. There is also an almighty god that can do everything. Many mortal beings pray to Administrator, but the religious people know that System is the omnipotent god.

The Good Thing™ of Windows model is that the almighty god can delegate special tasks to other minor gods by providing them with the adequate power. This powers are called user rights in the Windows model and there are many of them defined[8] so it can be achieved a high level of granularity in delegating tasks and privilege separation.

2.3.2 Existing Linux solutions

The most relevant solution available for Linux is the implementation of POSIX capabilities[9]. POSIX capabilities are also based in the unfinished POSIX 1003.1e draft[2].

The idea behind POSIX capabilities is to have some degree of separation of the different privileges that are needed in the system. There are 29 capabilities defined in kernel 2.4.22, the one used in Fedora Core 1. When these capabilities are assigned to an unprivileged process, they allow it to perform the different privileged tasks. This ability to set only the required level of privilege of a process results in a better control over what the different processes can do.

A correspondence can be established between the POSIX capabilities and the Windows rights. For example, the right to *debug programs* is equivalent to the capability CAP_SYS_PTRACE, or the right to *backup files and directories* is a subset of the CAP_DAC_OVERRIDE.

The idea as described in the capabilities man page[10] behind the draft of POSIX 1003.1e are:

- For every privileged operation the kernel checks the privilege sets.
- There is programming interface to access and set the privilege sets.
- There is support for saving the capability sets in the file system.

Only the first two have been implemented in Linux so far.

As part of the data that the kernel keeps for every process, there are three fields, named capability sets, that are bit masks with the capabilities of that process. Each bit of a capability set represents a capability that is enabled if the bit is set. The table that converts capabilities to bits and vice versa, as well as an explanation for each of them, can be found in Fedora Core 1 in the kernel source file `include/linux/capability.h`.

The explanation of the three capability sets is:

Effective contains the capabilities that are currently enabled and can be used by the process.

Inheritable indicates which capabilities will be preserved after a call to `sys_exec`.

Permitted describes the maximum set of capabilities that will be used by this process. If a capability is not present in this set, the process will not be able to use it, even if it was in the effective set. Also if a capability is not present in this set, it will not be passed to a process resulting of a `sys_exec` call, even if it was set in the inheritable set.

There is a also a system-wide limit to the capabilities that a process can use that is controlled by the kernel. It is called the capability bounding set. The initial value of the capability bounding set (`CAP_INIT_EFF_SET`) is defined in one of the kernel include files —`include/linux/capability.h`— and set by the kernel itself in `kernel/capability.c`. User processes can only drop capabilities from the bounding set, excluding `init` that can also add capabilities to it. Once the system administrator has dropped a capability from the bounding set it can only be re-added by restarting the system or through the `init` process.

Of course this mechanism is not perfect. Some of the capabilities can be used to gain access to the rest of them. As an example, if a process has the CAP_SYS_MODULE it can load a module that sets new capabilities and even modifies the capability bounding set. Other capabilities such as CAP_SYS_PTRACE or CAP_DAC_OVERRIDE can be used also to obtain the similar results. Windows privileges have the same problem though.

2.3.3 The missing parts

There are three ways in which POSIX capabilities can be used:

1. From a privileged process, programmatically drop every capability that is not needed from its capability set. Then, if a vulnerability of the privileged process is exploited and arbitrary code can be executed, the privileges used by this arbitrary code will be limited to those that are strictly needed by the privileged process. This will reduce the immediate impact of an attack.

   Sendmail takes advantage of capabilities in this way.

2. Assign privileges to the user when she logs in to the system. The user logs in to the system using a privileged process like `login`, `gdm`, `kdm` or `xdm`. A pluggable authentication module is then used that assigns capabilities —as stated in a configuration file that enumerates the capabilities assigned to every user— to the first process of the user for that session. The rest of the user processes inherit these capabilities.

   There is a module that does exactly this. It is called `pam_capability`[11] and comes with a patch that allows non-root users to pass on capabilities when they execute (`sys_exec`) binaries. However, this module has been inactive during almost two years and it is not included in any of the major distributions.

3. There is a capability set in the metadata of each file. This works similarly to the setuid bit. If a capability is set and the file belongs to root, the system creates a process with the capability enabled. The rules for setting this capability set will be the same ones that are used for dealing with the setuid.

   This is not implemented yet[12].

Sadly enough there is not much evidence of usage of the Linux capabilities. They are still included in the standard kernel, but all the information and programs have not been updated for years. Some utilities that are mentioned in many documents cannot be found anywhere.

2.4 Delegation of administrative tasks

Windows administrators can deploy custom snap-ins of the MMC to delegate tasks. I will review alternatives in the Linux arena.

2.4.1 Concept and usage

Windows system administrators work very often with the *Microsoft Management Console* (MMC). The MMC[13] is a only a framework for centralizing administrative utilities that can be added as snap-ins. These snap-ins remain seamless integrated thanks to the extensible, common presentation service provided by MMC.

Using the snap-ins and helped by wizards, the administrator can create custom management tools, that can be saved for later use, or for sharing with other administrators and users. The task of creating customized MMC consoles does not require any programming knowledge.

The new tools can have different levels of complexity so delegated tasks are not as complex or powerful as the ones performed by the administrator. The author mode of the MMC allows the administrator for an easier access and edition of the snap-ins.

Most often, a regular user requires additional rights or permissions to be able to use the custom MMC. For example, it would be worthless to create a custom console to run the disk defragmenter by a regular user if the administrator does not give this user the right to *perform volume maintenance tasks*. Thus, this idea has a close relationship with the one of administrative rights.

2.4.2 Existing Linux solutions

Other UNIX operating systems, like HP-UX have this functionality already built in. HP-UX has a tool that is called *System Administration Manager* (SAM) and that can be used to delegate administrative tasks. Using Restricted SAM regular users can access certain areas of SAM as specified in a per-user configuration file.

Although, Linuxconf[14], debconf[15] and Webmin[16] are also centralized administration tools that share some of the principles used in designing SAM, they do not allow for the delegation of administrative tasks.

Probably the most commonly deployed solution for delegating administrative tasks in UNIX is `sudo`[17]. `sudo` is a privileged program that allows regular users or groups to perform certain tasks as root. The tasks that each user can do are restricted by the sudo configuration file (`/etc/sudoers`) and can be controlled by user, machine, command and even options. Other than this, the user runs this processes with full privilege, so the administrator must choose very carefully which tasks can be delegated with sudo. If the task can, for example, spawn a shell, like `vi` or `emacs` can do, the *sudoer* will have complete control of the system. As an additional advantage, `sudo` does a very good job at logging everything.

2.4.3 The missing parts

It would be nice to have a program like sudo that would allow executing tasks with the same controlling criteria, but creating the new process with the required POSIX capabilities only.

There was a program to run other programs with a reduced set of capabilities that was called `sucap`, but I have not been able to find any copy of it. This program had a very different purpose than `sudo`, though. Instead of using a configuration file like the one for sudo, it read the capabilities from the command line and it could only be run by root.

2.5 Network domain

The topic in this section is the centralized authentication of users and computers within network domains.

2.5.1 Concept and usage

The relationship between computers and networks is closer everyday. If you have more than one computer in the same environment, then you probably want to use the same user and password for all of them and transparent access to resources in every other computer in that environment. As an administrator you also want centralized administration of the users, not only the authentication, but also other aspects such as define who can do what and where, or something as simple as an homogeneous audit report in which users have the same id independently of the system they used to login.

Windows has been using the domain concept for a long time now. It was mention for the first time in 1993 with the release of Windows NT 3.1, but it was not widely used until NT 4.0 was release in 1996. At that time, using a Windows domain meant centralized authentication of the users and computers and transparent access to the network resources. However, the biggest change to the concept was introduced with the release of Windows 2000 and the Active Directory Service. It introduced the ability to control other characteristics of the domain such as the DNS or the security policies that must be applied to each machine. But the Active Directory service is not just this. It constitutes a centralized configuration service for the whole network providing the infrastructure upon which other security enhancements are built. Features like the PKI service that comes with Windows 2000 strongly benefit from the Active Directory.

At the heart of the Active Directory there is a directory server with an LDAP interface and a hierarchical database engine that acts as the configuration repository, a Kerberos server to provide strong remote authentication and several tools —MMC snap-ins— and a lot of operating system mechanisms.

Most of the times, adding a new feature upon the Active Directory means modifying the schema. The schema defines what can be stored in the directory and it does so describing objects with attributes and relationships.

Administrators with a big number of systems, many users and need for some distributed security features will for sure appreciate having a domain service better than the yet too old NIS.

2.5.2 Existing Linux solutions

It probably makes no sense to look for something in UNIX that is exactly the same as the Active Directory in Windows. Mainly if we take into account that many of the features will need the basic infrastructure to get developed. So in this section I will describe solutions that provide the basic infrastructure: the directory service as a centralized authentication database. Since I am concentrating in UNIX based alternatives I will not cover using Linux clients in a Windows Active Directory domain[18].

There are many solutions available to provide the UNIX administrators with the same capabilities than their Windows colleagues in terms of centralized authentication. On the one hand, UNIX has been able to work with network domains since Sun introduced NIS in the mid-80's. Although, NIS and its evolution NIS+ are still in use in many production systems they are not the preferred option nowadays due to the lack of security of the former and the excessively complex administration of the later.

On the other hand, there are several commercial products that implement LDAP-based directory service and, even better, there is a free open source implementation of the directory services that is called OpenLDAP[19].

OpenLDAP is included in the Fedora Core 1 distribution. The distribution even allows for configuring LDAP authentication at installation time. Client side authentication requires the use of a PAM module (`pam_ldap`) that comes with Fedora Core 1 too.

There are very detailed explanations on how to configure OpenLDAP for authentication[20][21][22][24][23]. All of them provide information about the two biggest concerns from the security perspective: the fact that the passwords are transmitted in clear text and the requirement of restricting access to certain attributes using ACLs.

LDAP is an ASCII based protocol and it sends the passwords in clear text when trying to authenticate the user. This makes eavesdropping the password a trivial task. Nevertheless, OpenLDAP can use OpenSSL to protect the connections with strong encryption. A —self-signed or CA-signed— certificate must be installed and the LDAP daemon (`slapd`) must be configured to use it.

Some of the attributes are private, i.e. not everybody should be able to access them. The password is one of this attributes. The LDAP administrator must set access control lists to restrict access to this kind of attributes.

2.5.3 The missing parts

In the previous section I have talked about authenticating users. However, Windows domains do not only allow for user authentication; every system must be

able to authenticate to the domain before performing any user authentication. This functionality is not available in Fedora Core 1 or any other major distribution yet.

Anyhow, OpenLDAP authentication works, but it is not an out-of-the-box solution. It takes some work and a fair amount of knowledge to set up this centralized authentication system. More work is needed in automating this functionality.

2.6 Security templates

Windows comes with an application to check the security settings of the system against an externally defined set of good values and even change the ones that do not match the right values.

2.6.1 Concept and usage

Windows templates help the administrator to define, or import, a base line of the system, to verify the status of the system compared to the selected base line, and to even make the necessary changes to the system to enforce compliance to the base line. All this definitions are written in one single file that contains sections for authentication policies, audit policies, even logging, user rights, group membership, NTFS ACLs, registry ACLs and SACLs and service configuration.

Defining a security template can be a tedious task, but once it is done, it can be used for many systems that have the same or similar purpose. Even better, you do not have to start from scratch. There are many security templates defined by various organizations with a solid reputation in the security field[26][27].

Security templates are a great tool for hardening systems and auditing them periodically.

2.6.2 Existing Linux solutions

Bastille Linux[25] is a tool designed to harden Linux. It was originally oriented to Red Hat, but it supports many other operating systems now like Debian GNU/Linux and HP-UX.

Bastille includes only some of the functionality that can be found in the Windows security templates. It can use a predefined configuration file that can be designed once and used in many systems. It also does the job of applying the chosen configuration and even reverting the modifications to their original values. However, it is more oriented to change settings than to verify the configuration according to a base line and it does not have a verify-only mode.

It offers a graphical utility to edit the create the configuration settings that you want to apply to the systems. In contrast with the setting oriented configuration of the Windows security templates, this is more task oriented. It allows the user to decide whether to keep the setuid bit in mount explaining why it can be needed, instead of providing a list of allowed setuid programs. One of the strong points of Bastille over the Windows secure templates is that it has been designed to explain security issues to a system administrator. The system administrator can use this knowledge to take an educated decision about what should Bastille do.

No default configuration is provided with the program, thus Bastille must be run in interactive mode —with the command `bastille -x`— the first time it is used. Once the configuration file with the desired settings has been created, it has to be copied in every system by copying it to the expected path (`/etc/Bastille/config`) and then run Bastille in non-interactive mode with the `bastille -b` command.

2.6.3 The missing parts

The current release of Bastille (version 2.11) has not been updated to support Fedora Core 1 yet. It includes some information about Red Hat 9.0 and are working on Red Hat Enterprise Linux 3.0, though. Additionally, the perl-tk and perl-curses packages, that are needed to display the graphical or text interface, must be installed independently since they are not distributed with Fedora Core 1.

As I mentioned above, it should also have the ability to run in a verify-only mode. In this mode, Bastille would write a log with everything that doesn't match what is defined in the configuration file.

It would also be nice if the configuration editor allowed to define fine-grane policies too.

2.7 Group policies

I will discuss the advantages of being able to control different aspects of different systems within my domain as windows does with its group policy objects.

2.7.1 Concept and usage

It is common sense to apply the same security settings to systems that perform the same tasks. However, in some environments the usage of each system depends on the current user. I would want to have different security settings in the computer when it is used by a regular user that works on administrative tasks, than when an administrator is troubleshooting a problem from this system.

First, a centralized configuration tool is needed that stores, provides and changes upon request every configuration parameter of the system. I can read your mind now and you know what all this centralization of the configuration settings means: the registry. The Windows registry is the "central hierarchical database used in Windows to store information necessary to configure the system for one or more users, applications and hardware devices," as defined by Microsoft itself. Although the objective of centralizing the configuration of the system is clearly achieved, there are a number of aspects of Microsoft's implementation that many people dislike:

Cryptic Many of the keys are undocumented. It is sometimes difficult to know what a key does and which are the allowed values and their meanings.

Binary storage The registry is stored in a few files in binary format. The data stored in the registry can only be accessed with specialized tools. In case of emergency, it cannot be edited with a text editor.

Single point of failure If one application fails and corrupts the registry, the rest of the applications and the operating system itself that rely on it will fail.

Performance degradation As more and more programs are installed, the registry grows and access times are increased. This is also called the *registry creep* phenomenon and is one of the main causes for Windows systems slowing with age as software is added.

Lack of timing Because everything is stored in a few files, the user cannot use the create, modify or access times to find out what has changed if she has a problem.

After hearing about all these *features* you probably would prefer to avoid having something like the Windows registry in you UNIX systems. However, there should be something in the middle between having every configuration file under `/etc` with a different format and the Windows registry.

One of the main reasons for wanting to have a centralized configuration tool is that it could be used to apply Group Policies. Windows Group Policy Objects allow specifying settings such as registry values, log on and log off scripts, or software installation and maintenance options. These GPOs can then be assigned to any site, domain, or organizational unit defined in the Active Directory. When a user that belongs to one of these sites, domains or organizational units, logs in to the system, the settings contained in the GPO are applied, and when she logs off the changes are reverted.

2.7.2 Existing Linux solutions

Most of the times, configuring a UNIX computer means that the administrator logs in the machine and edits the required configuration files. Thus, the simplest way is to customize all the configuration in one system and copy or synchronize them with the rest. Obviously this method presents a lot of problems, like dealing with the different versions of the same applications or enforcing settings to the users without bloating their configurations.

Some tools like cfengine[28] try to solve the many formats problem by providing a higher level language to automate configuration and other administrative tasks which is platform independent while keeping the different configuration files used by each application.

Other implementations prefer to establish a common format and API for solving the configuration issues. Each of the main Linux desktops has its own: Gnome uses GConf[29] and KDE uses the Kiosk framework[30]. Both mechanisms provide the system administrators with ways to set default values as well as set and enforce configuration parameters for all the desktop applications. However, they are incompatible and neither a KDE application can use Gnome's configuration interface nor vice versa. It is possible that the D-Bus messaging system[31], which is currently used by the Gnome desktop, is also adopted by the KDE project improving compatibility in the future. They also do not import the configuration files of other applications that are not programmed to work with these interfaces.

There are many theoretical approaches[32] to create one common solution, but many of them fail to consider the distributed aspect and its security implications. Only a few take into account the importance of performing a secure authentication before allowing any changes[33].

2.7.3 The missing parts

There is a hot ongoing debate about modifying the way Linux is configured. Some people think that changing the way Linux applications are configured would mean changing the whole UNIX paradigm that lays underneath it. Some others think that we need everybody to agree in one centralized configuration interface that can be used by all programs.

The centralized configuration abstraction would have a lot of problems with the existing configuration files as many other projects have shown before[14][34]. But getting rid of them would imply migrating every application to the new configuration interface an loosing compatibility with all prior versions and other similar systems (like Mac OS X).

Finally, having the centralized configuration tool is not enough. There should be a mechanism to download the policy from the LDAP server in a secure manner and apply the settings as indicated by the policy. Probably this part should be implemented in a PAM module that is included in the session management group.

2.8 Trusted path

Windows uses the secure attention sequence (Control-Alt-Delete) to ensure the user that it is talking to a legit part of the operating system. This reduces the probability of being able to use a trojanized version of the log on application.

2.8.1 Concept and usage

Everything you do implies a certain amount of trust. For every decision you take, you do your own risk analysis analyzing information and trust.

Let me show you an example. Everything you eat could potentially kill you, but you eat it because:

- You have cooked it yourself.
- You know and trust the person who prepared the food.
- You do not know who prepared the food, but know and trust the restaurant or its owners.
- You have had previous and positive experiences in this place.
- You do not know the place at all, but you think it is not worth for them to loose a customer.
- You do not have a bad feeling about the food: it looks, smells and taste good.

- ...

Most of these points involve trust.

Thinks are much harder when we move from the real world to the computer world. When you login to a system, you implicitly trust the application you use to login. You tell your most precious secret, your user and password, to the application and you trust that it will not save it somewhere or sent it to somebody else. Obviously you trust it because *you believe* that this is a system application, and as such:

- You trust the company that wrote the operating system.
- It has been revised by many people.
- It looks exactly as the application that you have used many times to login to the system.

But could you differentiate this application from another one that looks the same? If you want to login to a UNIX workstation and you see the graphical login application —`xdm`, `gdm` or `kdm`,— how do you know that it is the genuine one instead of a full screen application being run by the previous user? Remember that the source code for many of these applications is available and it would be easy to create a modified version of that application. This application is called a Trojan horse: a program that seems to be a useful application and does something evil also.

Windows provides its users with a trusted path to help them to know that they are using the legitimate application and not a Trojan horse. When a user wants to log on to the system, she must press the Secure Attention Sequence which is the very well know key sequence Control–Alt–Delete[1]. This sequence cannot be intercepted by any application because it is directly answered by the operating system. Let us see exactly how this happens.

When the kernel has finished booting, it starts the session manager subsystem or SMSS. The session manager subsystem is like the init process in UNIX. It does many of the initialization tasks of the system and starts and controls the WinLogon process —one per Terminal service connection plus the one for the local system— and the Win32 subsystem or Client Server Runtime Subsystem (CSRSS). WinLogon is the process in charge of all the interactive user logons, but it does not have a hardcoded way to ask the user for her credentials —username and password. Instead of that, the abilities to identify and authenticate the user are implemented in the Graphical Identification aNd Authentication (GINA). The GINA is a dynamically loadable library (DLL) used by WinLogon that takes care of displaying the user interface to log on, passing the data to the Local Security Authority Subsystem (LSASS) and to create the shell, by default, `userinit.exe` that runs the Windows Explorer[36].

In an analogy with the XWindow System used in many UNIX machines, WinLogon corresponds to the display manager program (`xdm`, `gdm` or `kdm`) while the GINA

[1] Also known as the *three finger salute*.

acts similarly to a pam module. The main difference is that the GINA is responsible to display the window that asks for the user and password, while the pam modules are not.

As part of its initialization, WinLogon creates three desktops: the default desktop, the one that you normally uses, the winlogon desktop that is used to prompt the user for her password and can only be accessed by WinLogon, and the screen saver desktop. It also registers the secure attention sequence. If the secure attention sequence is detected by the kernel its keyboard hook handler gets notified, the user is switched to the winlogon desktop, and the authentication process takes place in a secure manner.

The fact that the secure authentication sequence is intercepted directly by the kernel and the trusted process WinLogon is invoked results in a trusted path to the user. However, you should take this solution for what is worth. Windows trusted path solution assumes that winlogon and the GINA can be trusted. It will not help if the intruder has already got administrative privileges somehow and the GINA has been replaced. Ed Skoudis' "Malware"[37] shows a good example of how this can be done and also explains the benefits for the intruder that already has the administrative privileges. Also, the trusted path solution does not work for remote logons.

2.8.2 Existing Linux solutions

There are several solutions available for Linux[35] and Fedora Core 1 implements them all.

If you login through one of the display managers, you can always restart the X server by pressing Control–Alt–Backspace. This will cause the server and all the graphical applications to be killed. The server will be restarted if the Fedora Core 1 system is in run level 5, and the display manager application will be the one configured by the system administrator or the distribution's default.

If the kernel has been compiled with system request (SysRq) configured and it is enabled there is a secure access key combination available. The status of the SysRq at run-time can be checked and set via the `/proc` interface. The kernel parameter is `/proc/sys/kernel/sysrq`. When enabled the key combination Alt-SysRq-k will kill all programs on the current virtual console.

For kernels newer than 2.4.3, the `loadkeys` utility, that is the one that loads the tables used to translate the key codes, has a SAK key action available. Executing the following command as root,

```
echo "control alt keycode 101 = SAK" | /bin/loadkeys
```

the secure attention key is generated whenever the user presses the Control–Alt–Pause keyboard combination as recommended in the kernel documentation referring to SAK[38].

2.8.3 The missing parts

The first two solutions, killing the X server and using SysRq, work in the default installation of Fedora Core 1. The solution of the SAK key action implemented in the kernel does not. Even though the functionality is included in the kernel, the default keymap does not assign this key action to any keyboard combination.

Although the solutions included in Linux could work, with different degrees of satisfaction, for the initial login, it will not work throughout the session. There is no way to ensure that the password given to the screen saver is directly asked by the operating system. Of course, you could still kill all the processes that you were running and loose some amount of work that was already done.

2.9 Automatic distribution of patches

It has been a long time since Microsoft realized that their customers do not patch their systems as often as they should. To try to get rid of the, sometimes well deserved, reputation that Windows systems are inherently insecure, they have put a lot of money and effort in the mechanisms used to patch Windows. As a result, Windows has several methods available for a faster and easier distribution of patches.

2.9.1 Concept and usage

As for most operating systems, patches can be downloaded from a website and installed manually. The usage of this site was almost restricted to the people who know that they have a problem or the very security conscious. Hence, they decided to offer the Windows Update service.

Windows Update is a web site (http://windowsupdate.microsoft.com/) that offers Microsoft customers the ability to automatically patch their systems. Everything the user needs is already included in the Windows operating system: the Internet Explorer web browser and a valid license number. When Internet Explorer accesses the website it downloads an ActiveX control that scans the system and sends information to Microsoft to let the server side decide which updates are needed. The user then accepts the updates and they are downloaded and installed.

Windows Update requires the user to connect periodically to the website and accept the updates. Maybe that is asking too much for a fair amount of Windows users, so they decided to include a built-in feature to do automatic updates. Automatic updates works like Windows Update but in unattended mode. Updates can be scheduled and the user gets notified before they are downloaded or installed.

The administrators of a Windows Active Directory domain have other solutions available to make sure that the systems are updated as required. Global Policy Objects (GPO) can be used to install software in msi or zap formats. While this is undoubtedly helpful, most if not all Microsoft patches and service packs do not come in these two formats.

The Software Update Service (SUS) is the evolution of the Automatic Update feature. Internal software updates distribution servers can be used, so that, in-

stead of having every system connecting directly to Microsoft and asking for updates, these servers download all the software updates and act as local repository. Administrators can select which updates are applied to the rest of the organization, and can even have different SUS servers to update different kinds of systems. The SUS server that is used by each system, as well as other parameters of the client, can be defined in a GPO.

2.9.2 Existing Linux solutions

In 1999, the Debian team released the version 2.1 of Debian GNU/Linux codenamed Slink. This release introduced a new tool written by Jason Gunthorpe which tremendously simplified the managing of dependencies of Debian packages and automatized the whole process of downloading and installing software. This tool is the Advanced Package Tool (APT)[39] and allows the automatic retrieval from the net of every package needed for a specific action. It can even upgrade the whole system to the next major release.

Other distributions tried to have the same functionality. Since Debian uses deb as their package format and all the other major distributions use rpm, some people modified APT to work with rpm.

Fedora Core 1 does not include APT with the official packages of the distribution. Red Hat introduced `up2date` as part of their Red Hat Network (RHN) subscription service, but it can be used with Fedora Core 1 without subscription to the RHN.

Updating the system with `up2date` can be as easy as executing the command `up2date-nox -u`. However, there is a very well known problem with Fedora Core 1 `up2date`, if you are not using a subscription to the RHN you will often get corrupted versions of the packages you are trying to download. `up2date` will not install the corrupted version of the RPM, but it will complain that the GPG signature of the package is not valid:

```
The package ... does not have a valid GPG signature...
```

Running `up2date` again will restart the process from the last good package. However, most of the times this problem is due to the excessive load handled by the Red Hat servers that have the packages available for downloading and `up2date` will return the same error over and over again. This behavior can be avoided replacing the URL of the Red Hat server in the sources configuration file used by `up2date` — `/etc/sysconfig/rhn/sources`— by any of the ones included in the following web page http://fedora.redhat.com/download/mirrors.html. This modification can also be used to have internal repositories for the updates instead of using the publicly available ones, thus reducing the external bandwidth consumption and having more control over what gets distributed.

There is another tool that comes with Fedora Core 1: `yum`. The Yellow-dog Updater Modified (YUM) provides a similar mechanism to the one offered by APT and works out of the box with rpm packages.

All this applications can be run as a cron job to comply with you security policy.

2.9.3 The missing parts

There is no really good solution for pushing patched software other than login to the every system and run the tool to update the system. If `up2date` runs often, three or four times a day, the amount of network traffic generated is very low when there is nothing new to be installed.

2.10 Integrated PKI and certificate management

Windows comes with everything you need to implement your own PKI already integrated in the system with the ability to manage certificates. This also permits using the encrypted file system and even smart cards. This is a whole topic in itself and I will only explain the high level stuff: the problem it solves and the parts involved. I will not get into any detail about cryptography[46].

2.10.1 Concept and usage

Public key, or asymmetric, cryptography means that two keys are required for secure communication. One is the private key, that, as its name implies, must be kept secret, and the other one is the public key, that, of course can be made public. Everything that has been encrypted with the public key can only by decrypted using the private key and vice versa. If you want to send a private message to somebody, you use her public key to encrypt it. She will be the only one that can decrypt the message. If you want to send a message that can be proved that it comes from you, you use your private key to encrypt the message. Everybody will be able to decrypt your message and will know that if comes from you because it was encrypted with your private key that nobody else knows. This is a rough description of the basis of public key cryptography.

The invention of public key cryptography changed the definition of some traditional problems in the world of secrets. You did not need to use a secure channel to get a shared key anymore. Now you could get somebody else's public key from any public place and establish a private communication with this person. The problem now was: how do you know that this key belongs to that person?

Maybe you could have got the key directly from her hands, but this cannot be the general case. There should be a way to know that the key that we are going to use is indeed the one of the person that we want to privately communicate with. If somebody that I trust could provide me with this confirmation, things would be a lot easier.

PKI means Public Key Infrastructure. And it is meant to solve the problem of key ownership. To achieve this goal it needs a set of standards for inter-operation that everybody agrees on, so different applications get to talk to each other and, most important, somebody you trust. The advantages of using a PKI is that I have and easy way to get a certificate with the public key of the person that I want to communicate with, and that this certificate is digitally signed by a *certification authority* (CA). The CA is an entity that deserves everybody's trust.

As you can see, a PKI does not only provide the organization with the ability to have public and private keys and do some crypto with them, but it is also responsible for the distribution of the certificates with the public keys, the secure distribution of everybody's private keys, handling the revocation lists, etc.

Certificates in windows are handled by the operating system itself. While this is a good thing because it standardizes the way applications and users deal with them, it also offers Microsoft an excellent opportunity to decide which certification authorities are honored by default and charge them for it.

The usage of Windows PKI in an organization provides stronger authentication methods (personal certificates and even smart cards), encrypted storage (EFS), encrypted and signed email, signed code, etc.

2.10.2 Existing Linux solutions

I have previously mentioned GPG signatures when talking about software distribution. *Gnu Privacy Guard* (GPG or GnuPG)[41] is a free implementation of public key encryption compliant with the OpenPGP standard[42], but it does not have the concept of a certification authority. Every GPG user can sign every body else's key and it is up to the GPG user to trust a key based on its signature. This is called the web of trust.

Fedora Core 1 comes with GnuPG and it is an essential part of the distribution because all the packages are signed with GPG. The default email clients (Evolution or KMail) can use GPG to encrypt and sign messages. KDE even has a utility for key management that is called KGpg. However, GPG is not compatible with TLS/SSL, so the GPG key cannot be used to authenticate the user for a web session. Also, it does not work with OpenSSH, although there is some work in progress[43] to solve that issue.

OpenSSL[44], also bundled with Fedora Core 1, offers a set of command line utilities that implement functionality to:

1. Create key parameters (RSA, DH and DSA).
2. Manage X.509 certificates, CSRs and CRLs.
3. Calculate Message Digests.
4. Encrypt and decrypt with ciphers.
5. Handle S/MIME signed or encrypted mail.
6. SSL/TLS client and server tests.

It is not integrated with the rest of the operating system and the documentation is not complete, but it is a very powerful toolkit. Many applications, like Apache, OpenLDAP or OpenSSH to name a few, depend on the OpenSSL libraries to offer cryptographic functionality.

2.10.3 The missing parts

Some functionality is missing in the solution provided by GPG: like smart card authentication or the ability to encrypt file system objects.

The average user would also need a wider support of GPG: more applications that can use GPG and better interoperability. The mozilla mail client cannot work with GPG out of the box. There is an extension called enigmail[45], but it is not part of the distribution. Also, Evolution and KMail implement message encryption in different ways: Evolution puts encrypted data in a MIME attachment and KMail encrypts the data in-line.

OpenSSL is very powerful but not very user friendly. Improving the documentation is a must.

2.11 Integrated IPSec

Windows comes with the ability to establish secure communications with other hosts by using the IPSec implementation that is integrated in the operating system.

2.11.1 Concept and usage

IPSec[47] is a security extension of the IP protocol —implemented on top of it at the host-to-host level (layer 3)— that provides cryptography based authentication and encryption services. As the rest of the family of protocols that constitute the essence of the Internet, it was designed for inter-operability and based in a public standard.

Three protocols are involved when talking about IPSec:

Authenticated Headers (AH) protocol [48] used to authenticate traffic.

Encapsulating Security Payload (ESP) protocol [49] used to encrypt and authenticate traffic.

Internet Key Exchange (IKE) protocol [50] needed to negotiate the session key if AH or ESP are used.

Windows systems can be easily configured to use the IPSec functionality that comes with the operating system. This is specially useful for establishing *virtual private networks* (VPN) with dial-in or ISP connected computers.

2.11.2 Existing Linux solutions

Linux 2.6 comes with this functionality already built in it and uses IPSec-Tools[51] for the user space configuration, but it is not yet distributed with Fedora Core 1. Soon, it will be distributed with Fedora Core 2. Additional information on how to use IPSec in Linux 2.6 can be found in the IPSec-HOWTO[54]

Fedora Core 1 comes with the 2.4.22 kernel and none of the IPSec solutions available for this kernel[52][53] are included in the distribution. FreeS/WAN distributes the user space utilities and precompiled modules in rpm format for Fedora

Core 1 (up to the last but one kernel release). Openswan offers source code in `tgz` format that can be compiled with your system.

Both implementations work well with Fedora Core 1. FreeS/WAN is very much easier to install, but the project has been recently discontinued. They even interoperate successfully with other implementations like the Windows one.

2.11.3 The missing parts

The software required to make the system talk IPSec exists already and works very well, but it must come bundled with the distribution for wider usage.

3 Conclusions

Linux has a lot of security features already available and there are a large number of projects trying to develop additional ones. By the very nature of open source, some of these projects will succeed and be incorporated to the main stream of Linux distributions, while a larger number will fail an disappear. This evolutionary system has provided us with a very powerful and full featured operating system so far, but there is plenty of room to make a more secure Linux system.

Some features, like software distribution, have traditionally been solved more successfully in the Linux world. MSI is the Microsoft approach to software packages. But most of them are still not ready for Linux.

The areas in which consolidated standards can be found are easier to implement. The problem comes for new areas specially when interoperability is a requirement.

Many of the features commented throughout this document are not included in the very well known commercial UNIX versions. The vendors that create this operating systems can also take advantage of this list and for once in their lifetimes learn something technical from Microsoft.

References

[1] Grünbacher, Andreas “Linux Extended Attributes and ACLs.” URL: http://acl.bestbits.at/ (23 Apr. 2004)

[2] Winfried Trümper “Summary about Posix.1e” 28 Feb. 2004. URL: http://wt.xpilot.org/publications/posix.1e/ (23 Apr. 2004)

[3] Grünbacher, Andreas; Schilling, Jörg “Linux and Solaris ACLs - Backup of Access Control Lists.” URL:http://www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/star-acl.html

[4] Saradiya. “GUI ACL Permission Manager.” 12 Mar. 2004. URL:http://www.ameba6.com/guiaclmanager/ (30 Apr. 2004)

[5] Dalessandri, Marco. “Advanced Permissions - konqueror.” 10 Mar. 2004. URL:http://www.kde-apps.org/content/show.php?content=11315

[6] Godinez, Javier “Secure Auditing for Linux.” Feb. 2003. URL: http://secureaudit.sourceforge.net/ (1 May 2004)

[7] Corbet, Jonathan. “The lightweight auditing framework.” Linux Weekly News. 8 Apr. 2004. URL:http://lwn.net/Articles/79326/ (2 May 2004)

[8] Microsoft Corp. “Chapter 4 - User Rights Assignment.” Threats and Counter-measures Guide. 2004. URL: http://www.microsoft.com/technet/Security/topics/hardsys/tcg/tcgch04.mspx (24 Apr. 2004)

[9] Tobotras, Boris. “Linux kernel capabilities FAQ.” v.2.0. 2 Apr. 1999 URL:http://www.kernel.org/pub/linux/libs/security/linux-privs/kernel-2.4/capfaq-0.2.txt

[10] Linux man pages. “Capabilities.” 23 May 2003. URL:http://annys.eines.info/cgi-bin/man/man2html?capabilities+7

[11] Kline, Erik. “Linux Capability PAM Module.” v.0.22. 11 Sep. 2002. URL:http://freshmeat.net/projects/pam_capability/

[12] Bacarella, Michael. “Taking Advantage of Linux Capabilities.” Linux Journal, issue 97 (May 2002): 72–75.

[13] Microsoft Corp. “Microsoft Management Console: Overview.” 7 Oct. 1999 ULR:http://www.microsoft.com/windows2000/techinfo/howitworks/management/mmcover.asp

[14] Glinas, Jacques. “Linuxconf – Linux administration made easy.” URL:http://www.solucorp.qc.ca/linuxconf/ (1 May 2004)

[15] Hess, Joey. “The many faces of Debconf”
URL:http://kitenet.net/programs/debconf/ (1 May 2004)

[16] Cameron, Jamie. “Webmin” v.1.140. 5 Apr. 2004.
URL:http://www.webmin.com/ (1 May 2004)

[17] Miller, Todd. “Sudo Main Page” v.1.6.7p5. 8 May 2003
URL:http://www.courtesan.com/sudo/ (1 May 2004)

[18] Ortiz, Jorge “Does Windows 2000 security model get along with my Linux?” 7 Dec. 2004
URL:http://www.giac.org/practical/GCWN/Jorge_Ortiz_GCWN.pdf
(1 May 2004)

[19] OpenLDAP Foundation. “OpenLDAP community developed LDAP software.” v.1.58 14 Dec. 2003
URL:http://www.openldap.org/ (1 May 2004)

[20] Bauer, Mick. “Paranoid Penguin: LDAP for Security, Part I.” Linux Journal, issue 111 (Jul. 2003): 34–37.

[21] Bauer, Mick. “Paranoid Penguin: Authenticate with LDAP.” Linux Journal, issue 112 (Aug. 2003): 30–33.

[22] Bauer, Mick. “Authenticate with LDAP, Part III.” Linux Journal, issue 113 (Sep. 2003): 32–36.

[23] Pinheiro-Malre, Luiz Ernesto. “LDAP Linux HOWTO.” v1.09, 5 Mar. 2004
URL:http://www.tldp.org/HOWTO/LDAP-HOWTO/ (1 May 2004)

[24] Danen, Vincent. “Using OpenLDAP For Authentication; Revision 2.” 6 May 2003.
URL:http://www.mandrakesecure.net/en/docs/ldap-auth2.php (1 May 2004)

[25] Lasser, John; Beale, Jay “Bastille Linux.”
URL:http://www.bastille-linux.org/

[26] The Center for Internet Security. “Benchmark Tools.” Mar. 2004.
URL:http://www.cisecurity.org/bench_win2000.html

[27] “Windows 2000 Security Recommendation Guides.” Release 1, Jun. 2001.
URL:http://intranet.logiconline.org.ve/Teach/W2K security/

[28] Burgess, Mark. “Cfengine - a configuration engine for Unix and Windows.”
URl:http://www.cfengine.org/ (24 Apr. 2004)

[29] Pennington, Havoc. “Introduction to the GConf library” 2000.
URL:http://developer.gnome.org/feature/archive/gconf/gconf.html
(24 Apr. 2004)

[30] Seigo, Aaron J. “KDE:The Korporate Desktop Environment.” Linux Magazine. Nov. 2002
URL:http://www.linux-mag.com/2002-11/kde_01.html (24 Apr. 2004)

[31] Pennington, Havoc. “D-BUS Tutorial.” v 0.1
URL:http://www.freedesktop.org/software/dbus/doc/dbus-tutorial.html
(24 Apr. 2004)

[32] Arnison, Matthew. “How to Fix the Unix Configuration Nightmare.” 16 Feb. 2002.
URL:http://freshmeat.net/articles/view/400/ (24 Apr. 2004)

[33] Long, Jason. “A Solution to the Problem of Configuration in Linux.” 28 Sept.2002.
URL:http://freshmeat.net/articles/view/565/ (24 Apr. 2004)

[34] Dagenais, Michel. “The Future of System Configuration” 23 May 2003.
URL:http://www.professeurs.polymtl.ca/michel.dagenais/pkg/config.html
(24 Apr. 2004)

[35] Wheeler, David A. “Secure Programming for Linux and UNIX HOWTO.” v 3.010, 3 Mar. 2003.
URL:http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/pdf/Secure-Programs-HOWTO.pdf (25 Apr. 2004)

[36] CSWL. “Windows NT/2000 Login Security.”
URL:http://www.cswl.com/whiteppr/white/gina.html (25 Apr. 2004)

[37] Skoudis, Ed. Malware. Fighting Malicious Code. Upper Saddle River, NJ: Prentice Hall. p. 347–352.

[38] Morton, Andrew. “Linux 2.4.2 Secure Attention Key (SAK) handling.” Kernel documentation. 18 Mar. 2001.
URL:http://www.linuxhq.com/kernel/file/Documentation/SAK.txt
(25 Apr. 2004)

[39] Noronha Silva, Gustavo; Mora, Hugo. “Apt HOWTO.” v.1.8.4 Apr. 2003.
URL: http://www.debian.org/doc/manuals/apt-howto/ (25 Apr. 2004)

[40] Vidal, Seth. “Yellow dog Updater, Modified.” 30 Jun. 2003.
URL:http://www.linux.duke.edu/projects/yum/index.ptml (26 Apr. 2004)

[41] Koch, Werner. “The GNU Privacy Guard.” v.1.55. 03 Jan. 2004.
URL:http://www.gnupg.org/ (26 Apr. 2004)

[42] Callas, J.; Donnerhacke, L.; Finney, H.; Thayer, R. “OpenPGP Message Format.” Internet Engineering Task Force, Nov. 1998.
URL: http://www.ietf.org/rfc/rfc2440.txt (2 May 2004)

[43] Weber II, Joel N. “OpenSSH-GPG.”
URL:http://www.red-bean.com/ nemo/openssh-gpg/ (2 May 2004)

[44] Engelschall, Ralf S. “OpenSSL.”
URL:http://www.openssl.org/ (2 May 2004)

[45] Brunschwig, Patrick. “Enigmail.” v.0.83.6.
URL:http://enigmail.mozdev.org/

[46] Xenitellis, Symeon. “The Open-source PKI Book” v.2.4.7 23 Jul. 2000.
URL: http://ospkibook.sourceforge.net/ (2 May 2004)

[47] Kent, S. & Atkinson, R. “RFC2401: Security Architecture for the Internet Protocol.” Internet Engineering Task Force, Nov. 1998.
URL: http://www.ietf.org/rfc/rfc2401.txt (2 May 2004)

[48] Kent, S. & Atkinson, R. “RFC2402: IP Authentication Header.” Internet Engineering Task Force, Nov. 1998.
URL: http://www.ietf.org/rfc/rfc2402.txt (2 May 2004)

[49] Kent, S. & Atkinson, R. “RFC2406: IP Encapsulating Security Payload (ESP).” Internet Engineering Task Force, Nov. 1998.
URL: http://www.ietf.org/rfc/rfc2406.txt (2 May 2004)

[50] Harkins, D. & Carrel, D. “RFC2409: The Internet Key Exchange (IKE).” Internet Engineering Task Force, Nov. 1998.
URL: http://www.ietf.org/rfc/rfc2409.txt (2 May 2004)

[51] Atkins, Derek. “IPsec-Tools.”
URL:http://ipsec-tools.sourceforge.net/ (2 May 2004)

[52] Gilmore, John. “Linux FreeS/WAN” v2.05 1 Mar. 2004.
URL: http://www.freeswan.org/ (2 May 2004)

[53] Many. “Openswan” v.1.0.3. 4 Apr. 2004.
URL:http://www.openswan.org/ (2 May 2004)

[54] Spenneberg, Ralf. “The official IPsec Howto for Linux.” v.0.9.6. 28 Jan . 2004
URL:http://www.ipsec-howto.org/ (2 May 2004)

Last Updated: July 25th, 2017

Upcoming Training

SANS Network Security 2017	**Las Vegas, NV**	**Sep 10, 2017 - Sep 17, 2017**	**Live Event**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**