Global Information Assurance Certification Paper

Operating System Build Management in the Enterprise

GIAC Security Essentials Certification (GSEC)
Practical Assignment
Version 1.4b

Duncan C Beattie

February 07, 2004

Table of Contents

Table of Figures

Abstract

Mitigating the risk to critical systems from vulnerabilities in operating system builds is an important responsibility of any system administrator. In organisations with a large number of servers, running multiple applications and services, managing the state of production builds can be a time consuming exercise. Regardless of the size of an organisation's IT infrastructure, it is crucial from a security perspective that administrators know the detail of the production operating system configuration on the servers for which they are responsible.

Establishing a standard system build policy for each operating system is the foundation upon which to build an understanding of systems, improving the ability to detect change and to understand the risk posed by new threats. This paper discusses points to consider in creating system build policies and how to tackle both bringing systems into compliance and ensuring that they remain compliant.

Intended Audience

This paper assumes an understanding of system administration. Awareness of the basic principle aims of IT security – to protect data integrity, availability and confidentiality – and an understanding of risk awareness and management is assumed.

The Case for Build Policies

"We need to work with companies to help them build security plans."
- Mike Nash, vice president of the security business unit at Microsoft [1]

Protection of business data and the availability and performance of online systems are key responsibilities of the system administrator. In a large environment an administrator's confidence in the servers he looks after will be dictated by the extent to which he knows his systems. If servers are released in an ad hoc manner, with the level of security in place dependant upon the individual carrying out the work, this confidence will be diminished. By implementing hardened systems according to a technical build policy, and monitoring build compliance post-release, the security of the environment can be greatly enhanced.

Without build policies that cover secure baseline settings and detail patching requirements and intervals, the vulnerability to new risks can be greater. Furthermore, the ability to detect signs of intrusion is also greatly enhanced by establishing policies that define a normal system state.

Policy Definition and Documentation

Information security policies underpin the security and well being of information resources.. they are the foundation, the bottom line, of information security within an organisation [2].

Security policies can be defined at many levels. Many organisations will have an umbrella corporate security policy that defines what is expected of staff that use company computers. High level company policies may contain information from fraud and money laundering prevention to CCTV (Closed Circuit Television) usage and business continuity guidelines.

To compliment physical security and regulatory guidelines there will also be rules governing the protection of information. Such information security manuals may define organisation practice including Internet usage, password standards, and corporate email policy in a manner that will be understood by all, from data entry clerks to IT operations staff.

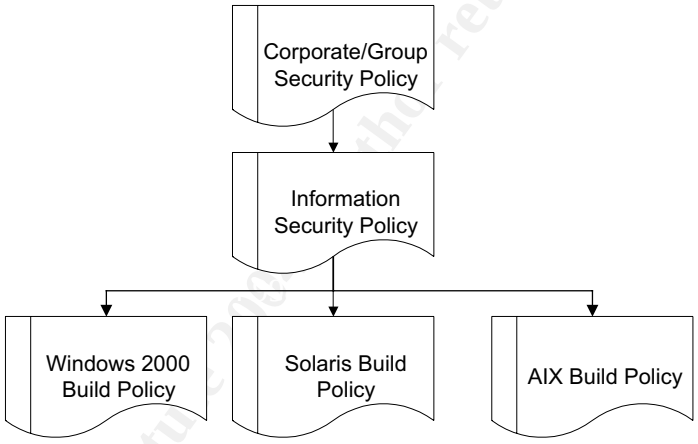

Figure 1: Security policy hierarchy.

Operational business security policies that are defined and published at a high level in the organisation serve as the foundation for information security policy, which in turn serve as the basis from which to develop technical build policies. It is the technical build documentation that serve as the baseline reference for putting policy into practice.

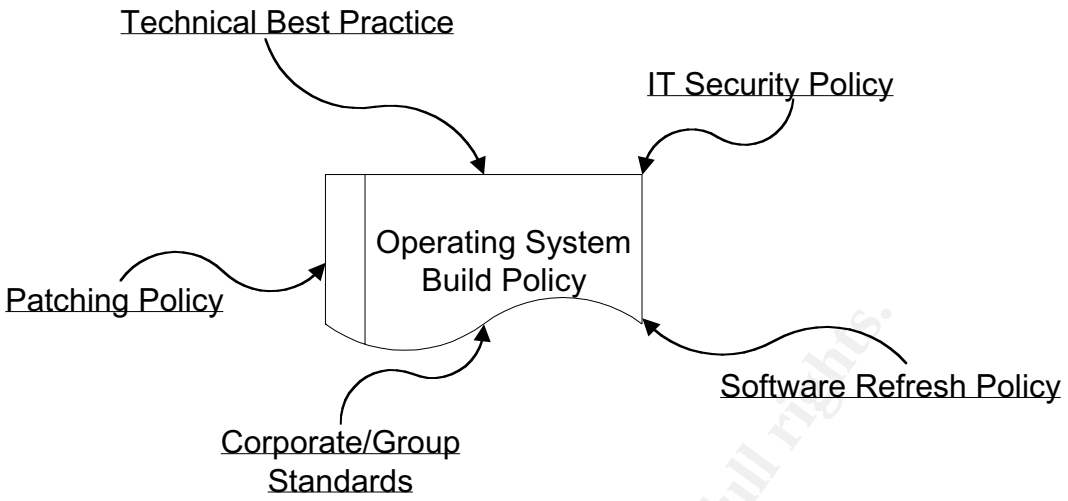

Figure 2: Build policy influences.

In addition to corporate security policy manuals, technical (or industry) best practice will influence the specific recommendations of an O/S build specification. There are many excellent resources available published online and in text books that describe hardening techniques for hardening Windows, Unix and Networking environments. Technical advice in hardening white papers and security manuals are also a good source of material that can be tailored for inclusion in each policy.

Each secure operating system build policy will document the technical configuration of a server build. The prescriptive configuration and settings for the following elements of a hardened build will be included:

- relevant subsystems - daemons - filesystems
- users and groups - password settings - file permissions
- privileged account access - auditing and accounting

Whilst a build policy may have a lot in common with system hardening instructions, the main difference is that it will define a baseline system state; the standards by which all servers must be built.

There are, of course, many different uses to which a production server may be put. Depending on their application, the baseline build of a server may differ. For example, the build of an Internet-facing web server will clearly bear little relation to that of an internal data server. In order to understand the environment an administrator oversees, it will often be the quality and relevance of his documentation that makes build management a less arduous task.

A documented technical system build policy can be used in such bread-and-butter administration work as preparing a server build before it is introduced to the production network. Similarly, the standards defined for newly built boxes may be introduced to legacy systems that may not yet reflect the same levels of security.

System administration is rarely as clear cut as this, of course. There will often be circumstances where best practice cannot be followed without compromising the systems' ability to provide service to the business. Extra ports may need to be opened, incoming connections set up, network shares declared or additional privileges granted to user accounts. For example, the default standard for a hardened production server may include turning off X services. Whilst it is possible to temporarily open X ports on a server for the duration of an Oracle upgrade, some application administration tools – such as FileNet's Image Services System [3] – may require X access on a permanent basis. Just as it is important to document a standard for system builds, it is equally important to record exceptions to the norm.

Deviations from the documented baseline configuration can introduce elements that may be considered managed, acceptable risks depending on the circumstances. There are often ways of minimising the risk posed by leaving non-standard ports open, by front-ending daemons using tcp-wrappers to minimise access to specified users or IP addresses, or by tunnelling X communications over encrypted SSH connections. It is important to record the differences from standard baseline settings, as this information is crucial to understanding the strengths and weaknesses of any environment, and in defining a baseline for that system.

Implementing Standard Builds

When a technical operating system build policy has been produced, the administrator uses this as their specification for configuring new servers before they are released into production. The policy will also be used in scoping hardening exercises that are carried out on legacy environments to address any gaps in the state of older servers.

In large enterprises the volume of server builds and migrations often necessitate the use of baselined 'golden' server images, or scripts that automate the deployment of baseline servers. The use of such tools and techniques relieve the administrator of what can be resource-hungry work, but in terms of build management the main advantage is in the ability to deliver consistent, secured servers to the environment.

Administrators often develop scripts, small utilities and batch files to automate repetitive tasks. Operating system and application software, service pack installations, patching, environment variable settings, daemon configuration, backup and job scheduling settings, registry tweaks, ACL definition and of course security settings defined in the build document can all be set in automated build scripts.

To develop in-house scripts from scratch may not suit all enterprises. There are many packages and toolkits available that provide automated deployment and configuration facilities. These technologies include the following:

Ignite-UX	For automating deployment of HPUX servers.
BladeLogic	Compatible with multiple UNIX, Windows and Linux platforms.
Veritas OpForce	For rolling out consistent Windows, Solaris, AIX and Linux builds.
ProLiant Essentials Rapid Deployment Packs	To enable automated deployment of Windows or Linux on ProLiant servers.
System Preparation (Sysprep) tool, and Remote Installation Services (RIS).	Enabling automated installation of Windows Server 2003.

To illustrate the value of using tools that automate the deployment of standardised server images, the next section briefly describes how this can be done using Solaris Jumpstart.

Automating Solaris Deployment

Automation of Solaris deployment across multiple machines is provided by Solaris Jumpstart software. Jumpstart runs on a central server that contains operating system and application software in addition to configuration build scripts.

Jumpstart infrastructures are comprised of three core server components that may be provided by one or more servers. These are the boot, profile and install services. The boot service provides the IP addresses of the client, and Jumpstart profile and install servers using DHCP or RARP. The profile service provides the configuration information the install will refer to. As a minimum, details such as system locale, time zone, terminal type, security policy, name server and time service must be defined to enable an install to proceed without interaction.

The Jumpstart install service provides the Jumpstart client with rules that determine which scripts are run pre- and post- O/S installation, which profile is used and so forth. Within the base jumpstart directory there are further filesystem partitions that contain various categories of scripts, policies, operating system and application software.

Jumpstart directory structure:

/jumpstart/Begin
/jumpstart/Drivers
/jumpstart/Finish
/jumpstart/OS
/jumpstart/Packages
/jumpstart/Patches
/jumpstart/Profiles
/jumpstart/Sysidcfg

The install process uses a master `rules.ok` [4] file that includes one or more rules to determine how a target machine is installed and configured. Keyword parameters in the rules file that match system attributes such as hostname, network or machine type will determine which profile and which scripts are used during the install. For example:

```
network 152.3.140.0  profiles/trusted   scripts/trusted_finish
```

Here, a host on the 152.3.140 network will install Solaris using the profile `trusted` and finish the installation by executing the `trusted_finish` script. The finish script may be stored in the Drivers directory, which will in turn call further finish scripts. Driver scripts are commonly used in the rules file to call the desired start and finish scripts.

The sysidcfg file used during the installation process defines all of the necessary environmental settings such as networking details, time zone and root password attributes.

Jumpstart Installation Process

The following six steps illustrate the procedure of installing Solaris on a client machine using Jumpstart [5].

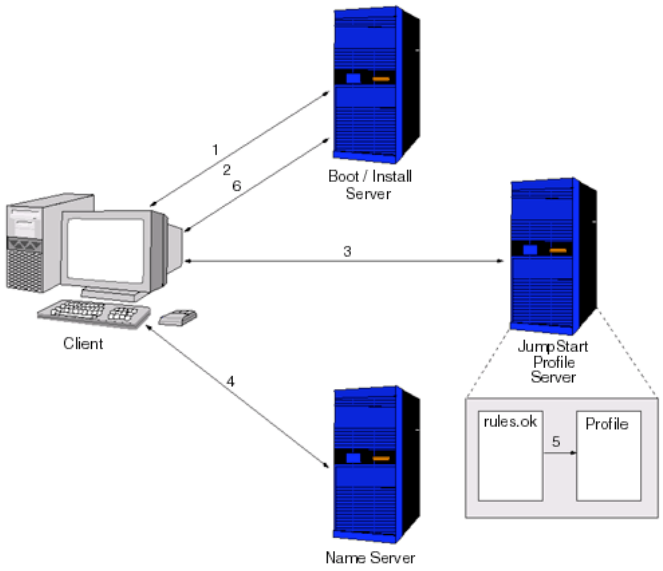

Figure 3: Jumpstart Installation [6]

1. First the client is turned powered on and boots as a network install. The client sends a RARP broadcast to determine its IP address to which the boot server responds with the necessary network interface data.
2. Having established the network information the boot process continues, loading the install kernel and the installs the operating system. It also determines the network information for the designated profile server containing a rules.ok file.
3. The sysidtool determines default information for the client in the name service.
4. The installation software contacts the profile server and searches for an appropriate rule in the rules.ok file for the client.
5. The rule for the client contains the name of the profile file to be used for the installation.
6. The installation software installs Solaris on the client according to the instructions in the profile file.

Jumpstart Architecture and Security Scripts

Solaris released the Jumpstart Architecture and Security Scripts (JASS) [7] toolkit as a means of implementing the hardening recommendations made in security articles at the Sun BluePrints web site [8]. The JASS toolkit works in unison with Jumpstart technology by supplementing profiles with finish scripts that carry out best practice server hardening and baselining.

The heart of the JASS toolkit is in the Driver scripts it supplies. These scripts called by the toolkit driver scripts are finish scripts only. The driver scripts contribute to the hardening process by defining environment variables used during the install, copying files to the client and defining the finish scripts that are run.

At the core of the toolkit is the driver.run script which takes the information fed to it from the earlier driver.init script (package mount directory, scripts directory, JASS suffix that is used when copying off replaced files) then proceeds to verify, commence and eventually conclude the installation process.

First, driver.run will verify the various mount points and host details defined in the environment variables. If any fail verification the process will abort and the installation will halt. The package, patch and any other Jumpstart filesystems required are then mounted on the client system. Having mounted the necessary directories, the files specified in the finish script are copied to the client. The scripts defined in the finish driver script are then executed, the Jumpstart filesystems unmounted and the client rebooted. Further driver scripts config.driver, hardening.driver and secure.driver assist in the install and hardening process by running initialisation tasks, defining the base OS install details, hardening script used during the install.

The raison d'être of the JASS toolkit is, after all, to assist in securing and baselining Solaris systems. To this end, the toolkit includes a comprehensive set of finish scripts that disable services, enable extra logging, install recommended patches and harden file permissions. These are listed in the appendix A.

Ongoing Build Maintenance

Having produced a document declaring secure build standards, and having introduced automated methods of deploying fresh server builds that adhere to them, it is important to ensure that the released level of security is maintained over time.

Introducing build standards in a large environment is best approached in an incremental manner. This is particularly true of hardening the legacy environment, but will eventually hold true for all servers as new standards are introduced and policies are revised. Initial build policy may only define a limited set of basic security measures, to address vulnerabilities or weaknesses that are trivial to resolve, for example.

When a server is released, it makes sense for it to be tested by IT security staff who can assess build compliance and sign off new installations. Once in production, every server will be subject to daily administration, development, project and architectural influences. The main security drivers for changes are in applying patches and workarounds to any vulnerability a vendor may publish.

Regular reviews and audits of system builds (often the responsibility of IT Security or Audit staff) are worth scheduling. Should any lapse in security be identified, remedial action may be taken to address any weakness. With effective basic monitoring procedures, weaknesses may also be investigated and addressed retrospectively soon after they are introduced. This iterative process of revising and implementing standards, and ongoing compliance monitoring is illustrated below.

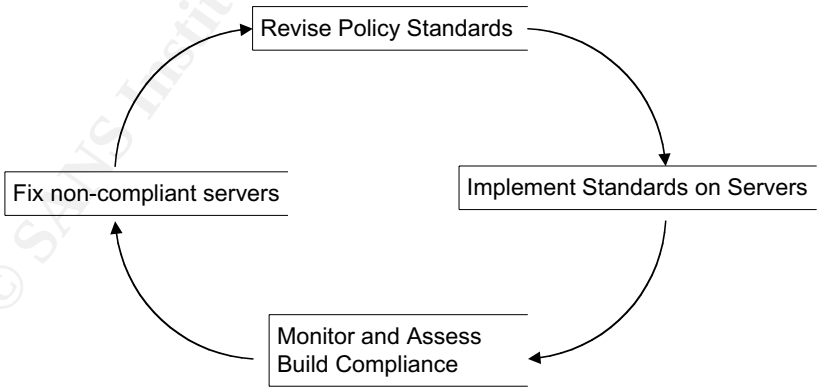

Figure 4: Build Maintenance Life Cycle.

Monitoring build compliance must cover all aspects of the policy that has been put in place. Tools such as nmap can be used to check that the expected network ports are listening. In house developed scripts can be used to report patch levels, and other system state information. This can be a very resource intensive and costly operation, indeed many firms opt to outsource security monitoring to third parties [9] or introduce intrusion detection to the infrastructure.

Running vulnerability scanners against systems will merely confirm that they are not vulnerable to known common threats, and that hosts do not exhibit the basic weaknesses that the policy was designed to mitigate. Tools such as nmap, Tripwire and Nessus are, along with central log monitoring, very useful additions to the defence-in-depth strategy of any IT security procedures. In an enterprise scale organisation it is worth investing in an enterprise scale solution for build compliance monitoring.

ISS Security Scanner, VigilEnt Policy Center and BindView policy compliance tools offer excellent solutions to monitoring multiple system builds to ensure that they remain in line with policy.

Enterprise Security Manager (ESM) from Symantec is a tool in the same class. The following section describes how ESM works in principle and gives an indication of the value of such tools to organisations that need to maintain large heterogeneous environments.

Build Policy Compliance Monitoring using Symantec ESM

To begin with, it is worth clarifying a few terms in relation to ESM. ESM *policies* consist of a number of *modules* that in turn are made up of a set of related *checks*. For example, there are modules that carry out checks on Password Strength, Network Integrity, OS Patches and Account Integrity. The extent to which a check can be tailored depends on the nature of the check. Checks that report whether unsuccessful SU attempts are logged or whether a server has shared folders allowing Full Control to Everyone configured may only be set on or off. Other checks have greater granularity, from defining the maximum number of characters in a password to listing users and groups that may log on locally, or have the ability to take ownership of files or other directories.

In some modules, checks on specific file properties and permissions, or registry settings, will compare values on target servers with those defined in an ESM *template*. Templates are used in the Registry, File Attribute and File Watch module checks, and to determine the patch sets installed on servers.

The configuration details of policies and registration details of *agents* are contained in a proprietary database on the ESM *manager* server. Agent software installed on target servers is configured to register with the manager. Console software allows remote workstations to connect to the manager to administer policies and templates, organise related agents into domains and view and manipulate if necessary the results of policy runs.

A policy run can be initiated by dragging a policy onto a domain or agent. Policy runs may also be scheduled using the scheduling functionality ESM provides. For example, to obtain data on the state of the network settings on Windows 2000 servers, a policy run may be scheduled to run a Network Integrity module on the Windows 2000 domain.

Creation of Revised Domains

ESM provides a number of default domains into which agents may be categorised. These are Windows 2000, Windows NT, UNIX, NetWare/NDS, and OpenVMS.

The creation of domains that segregate server by operating system, and by server type in the case of the larger NT environment, allow policy runs to be targeted more efficiently against related host populations.

Creation of Revised Policies

As a build compliance tool, in order for the data to be meaningful in any organisation, the settings on modules and checks need to reflect the standards that ought to be in place in the server O/S configuration. The security settings available in ESM policies detail very specific low level settings in addition to things like password settings that can be directly transposed from the guidance in build documentation.

Default polices are provided by ESM to get the security administrator up and running quickly. In order for ESM monitoring to be as relevant as possible it is worth developing an ESM policy (and associated templates) for each documented O/S build policy.

It is important that the settings in the policies are right. It is worth investing time in reviewing each and every check that is available is each module for each of the main O/S environments. Where appropriate, each check should be set up according to it's bearing to defined and published standards (documented or otherwise). Having established the initial state of the policies, it is also worth recording the settings in spreadsheets that include a cover sheet with version control data, which will be maintained on an ongoing basis as a record of their development and current status.

Build Compliance Procedures.

Having configured the build compliance tool to reflect the standards that have been implemented, the Audit and IT Security functions of an organisation can agree compliance strategies that make the most of the tool's features.

Full system audits can be carried out on a twice annually or quarterly basis. Smaller, more focussed policies can be scheduled to run on a weekly or daily basis to examine system compliance in critical areas such as listening TCP services, open shares and changes in access privileges. Compliance tools can be used to confirm that critical patches have been applied on all systems as evidence that systems adhere to patch policy and are not vulnerable to the latest threats.

They can also play a useful role in checking that audit recommendations have been carried out. For example, a template can be developed to test that file permissions on specific files and directories are at a prescribed level, that a file contains a given value or that a specific file exists. The compliance check that identifies a weakness can also be used to confirm that it has been resolved following a fix-up change.

Escalation procedures will need to be set up so that when weaknesses and non-compliant settings are found, they can be communicated at the appropriate level to the relevant administration and support staff. By feeding into an organisation's Risk and Compliance functions, effective use of compliance monitoring tools and techniques not only aid in identifying security weaknesses in system settings, but crucially, are key to providing evidence that standards *are* being met.

Summary

There is no point in any organisation developing well-intentioned, well-informed policies to govern the way IT infrastructure should be used and controlled, without implementing them at all levels. Equally, there is no benefit in developing automated systems to churn out clones of securely built servers, without putting measures in place to ensure they remain secure post-deployment. Regardless of the tools and technologies used in monitoring compliance, it ultimately depends on the quality of secured systems, the relevance of the governance controls, and the effectiveness of monitoring and escalation procedures to deliver a secure environment.

By defining standards, with the ability to provide evidence that they are maintained, the business will be in a sound footing from which to react to change, regulatory demands and future threats.

Appendix A – JASS Finish Scripts

For details on the function of each of these scripts, refer to JumpStart Architecture and Security Scripts for the Solaris Operating Environment - Part 3 By Alex Noordergraaf, [http://www.sun.com/blueprints/0900/jssec3.pdf].

Disable Finish Scripts

disable-asppp.fin
disable-autoinst.fin
disable-automount.fin
disable-core-generation.fin
disable-dmi.fin
disable-dtlogin.fin
disable-keyserv-uid-nobody.fin
disable-lp.fin
disable-nfs-client.fin
disable-nfs-server.fin
disable-nscd-caching.fin
disable-power-mgmt.fin
disable-preserve.fin
disable-remote-root-login.fin
disable-rlogin-rhosts.fin
disable-rpc.fin
disable-sendmail.fin
disable-slp.fin
disable-snmp.fin
disable-spc.fin
disable-syslogd-listen.fin
disable-system-accounts.fin
disable-uucp.fin

Enable Finish Scripts

enable-32bit-kernel.fin
enable-bsm.fin
enable-ftp-syslog.fin
enable-inetd-syslog.fin
enable-priv-nfs-ports.fin
enable-rfc1948.fin
enable-stack-protection.fin

Install Finish Scripts

install-at-allow.fin
install-cron-allow.fin
install-fix-modes.fin
install-loginlog.fin
install-newaliases.fin
install-openssh.fin
install-recommended-patches.fin
install-security-mode.fin
install-strong-permissions.fin
install-sulog.fin

Minimize Finish Script

minimize-iplanet-enterprise-server.fin

Remove Finish Script

remove-unneeded-accounts.fin

Set Finish Scripts

set-login-retries.fin
set-rmmount-nosuid.fin
set-root-password.fin
set-system-umask.fin
set-term-type.fin
set-tmpfs-limit.fin
set-user-password-reqs.fin
set-user-umask.fin

Update Finish Scripts

update-at-deny.fin
update-cron-deny.fin
update-inetd-conf.fin

References

[1] Lemos, Robert. "Decoding the lessons of Slammer" 4 March 2003. http://news.com.com/2008-1082-990757.html

[2] The Information Security Policies and Standards Group. URL: http://www.information-security-policies-and-standards.com/

[3] FileNet Image Manager – product literature. URL: http://www.filenet.com/English/Products/Datasheets/023250005.pdf

[4] Duke University Computer Science Department. "Duke Sysadmins Talk: Solaris Jumpstart." 6 July 2001. URL: http://www.cs.duke.edu/~braun/jumpstart/rules2.html

[5] Dubrawsky, Ido. "Jumpstart for Solaris Systems, Part One". 12 March 2001. URL: http://www.securityfocus.com/infocus/1383

[6] Kasper, Paul Anthony & McClellan, Alan L. Automating Solaris Installations. Prentice Hall / SunSoft Press, 1995.

[7] Sun Microsystems, Inc. "Solaris Security Toolkit (JASS)." URL: http://wwws.sun.com/software/security/jass/

[8] Sun Blueprints Online. June 1999 - January 2004. URL: http://www.sun.com/blueprints/browsesubject.html#security

[9] Faile, Jonathan S. "Security Outsourcing." 25 August 2001. URL: http://www.sans.org/rr/papers/index.php?id=223