the time the password was last changed, the expiration time as well as the validity period of the password and some other information that are not subject to the security functions as defined in this Security Target. Users are allowed to change their passwords by using the passwd command. This application is able to read and modify the contents of /etc/shadow for the user's password entry, which would ordinarily be inaccessible to a non-privileged user process (this implies that the TSF does not rely on the strength of the hashing algorithm to protect the passwords). Users are also warned to change their passwords at login time if the password will expire soon, and are prevented from logging in if the password has expired.

The time of the last successful logins is recorded in the /var/log/lastlog file.

The TOE displays informative banners before or while users are logging in. The banners can be specified with the files /etc/issue for log ins via the physical console or /etc/issue.net for remote log ins, such as via SSH. When logging into through the physical console, the banner is displayed above the username and password prompt. For logging in via SSH, the banner is displayed to the remote peer before the SSH-session handshake takes place. The remote SSH client will display the banner to the user. When using the provided OpenSSH client, the banner is displayed when the user instructs the OpenSSH client to log into the remote system.

This security function covers the SFRs of FTA_TAH.1, FMT_SMR.1, FTA_TAB.1, FAU_UAU.1, FAU_UAU.6, FIA_SOS.1, FIA_UID.1, FAU_GEN.1, FIA_AFL_EXT.1, FIA_ATD.1, FIA_USB.1, FIA_UAU.7, FTA_MCS.1.

7.1.1.4 User session handling

Sessions can be locked by users voluntarily via the screen application. In addition, the vlock application is started to protect the user's session after a configurable duration of inactivity on that session. To ensure that the session is always locked even when applications take full control of the session, a helper-daemon may be used that controls each session and terminates offending applications.

This security function covers the SFRs of FTA_SSL.1, FTA_SSL.2.

7.1.2 Audit

The Lightweight Audit Framework (LAF) is designed to be an audit system for Linux compliant with the requirements from Common Criteria. LAF is able to intercept all system calls as well as retrieving audit log entries from privileged user space applications. The subsystem allows configuring the events to be actually audited from the set of all events that are possible to be audited. Those events are configured in a specific configuration file and then the kernel is notified to build its own internal structure for the events to be audited.

7.1.2.1 Audit functionality

The kernel interface which provides the means to configure the audit properties is usable only by root users. Only processes possessing the root authority or kernel functions can submit audit records to the kernel which in turn forwards the audit records to the audit daemon. The audit daemon writes the audit records to the audit trail. An internal queuing mechanism is used for this purpose. When the queue does not have sufficient space to hold an audit record the TOE switches into single user mode, is halted or the audit daemon executes an administrator-specified notification action depending on the configuration of the audit daemon. This ensures that audit records do not get lost due to resource shortage and the administrator can backup and clear the audit trail to free disk space for new audit logs.

Access to audit data by normal users is prohibited by the discretionary access control function of the TOE, which is used to restrict the access to the audit trail and audit configuration files to the system administrator only.

The system administrator can define the events to be audited from the overall events that the Lightweight Audit Framework using simple filter expressions. This allows for a flexible definition of the events to be audited and the conditions under which events are audited. The system administrator is also able to define a set of user IDs for which auditing is active or alternatively a set of user IDs that are not audited.

The system administrator can select files to be audited by adding them to a watch list that is loaded into the kernel.

7.1.2.2 Audit trail

An audit record consists of one or more lines of text containing fields in a “keyword=value” tagged format. The following information is contained in all audit record lines:

- Type: indicates the source of the event, such as SYSCALL, FS_WATCH, USER, or LOGIN
- Timestamp: Date and time the audit record was generated
- Audit ID: unique numerical event identifier
- Login ID (“auid”), the user ID of the user authenticated by the system (regardless if the user has changed his real and / or effective user ID afterwards)
- Effective user ID: the effective user ID of the process at the time the audit event was generated
- Success or failure (where appropriate)
- Sensitivity label of the subject that caused the event

This information is followed by event specific data. In some cases, such as SYSCALL event records involving file system objects, multiple text lines will be generated for a single event, these all have the same time stamp and audit ID to permit easy correlation.

The audit trail is stored in ASCII text. The TOE provides tools for managing ASCII files that can be used for post-processing of audit data. These tools include:

- less - reads the ASCII audit data
- ausearch - allows selective extraction of records from the audit trail using defined selection criteria
- sort - The audit records are listed in chronological order by default. The sort utility can be used together with ausearch to use a different sorting order.

The audit trail is stored in files which are accessible by root only.

This security function covers the SFRs of FAU_GEN.1, FAU_GEN.2, FAU_SEL.1, FPT_STM.1. FAU_STG.1, FAU_SAR.1, FAU_SAR.3, FAU_STG.3, FAU_STG.4.

7.1.3 Discretionary Access Control

The general policy enforced is that subjects (i.e., processes) are allowed only the accesses specified by the policies applicable to the object the subject requests access to. Further, the ability to propagate access permissions is limited to those subjects who have that permission, as determined by the policies applicable to the object the subject requests access to.

A subject with a file system user ID of 0 is exempt from all restrictions of the discretionary access control and can perform any action desired. For the execution of a file by root, the permission bit vector of that file must contain at least one execute bit.

DAC provides the mechanism that allows users to specify and control access to objects that they own. DAC attributes are assigned to objects at creation time and remain in effect until the object is destroyed or the object attributes are changed. DAC attributes exist for, and are particular to, each type of named object known to the TOE. DAC is implemented with permission bits and, when specified, ACLs.

The outlined DAC mechanism applies only to named objects which can be used to store or transmit user data. Other named objects are also covered by the DAC mechanism but may be supplemented by further restrictions. These additional restrictions are out of scope for this evaluation. Examples of objects which are accessible to users by cannot be used to store or transmit user data are: virtual file systems externalizing kernel data structures (such as most of procfs, sysfs, binfmt_misc) and process signals.

During creation of objects, the TSF ensures that all residual contents is removed from that object before making it accessible to the subject requesting the creation.

This security function covers FDP_RIP.2 FMT_REV.1(1), FMT_REV.1(2), FMT_MSA.1(1), FMT_MSA.1(2), FMT_MSA.2, FMT_MSA.3(1).

7.1.3.1 Permission bits

The TOE supports standard UNIX permission bits to provide one form of DAC for file system objects in all supported file systems. There are three sets of three bits that define access for three categories of users: the owning user, users in the owning group, and other users. The three bits in each set indicate the access permissions granted to each user category: one bit for read (r), one for write (w) and one for execute (x). Note that write access to file systems mounted as read only (e. g. CD-ROM) is always rejected. Also, write access to file system objects marked as immutable is always rejected. The SAVETXT attribute is used for world-writeable temp directories preventing the removal of files by users other than the owner.

Each process has an inheritable “umask” attribute which is used to determine the default access permissions for new objects. It is a bit mask of the user/group/other read/write/execute bits, and specifies the access bits to be removed from new objects. For example, setting the umask to “002” ensures that new objects will be writable by the owner and group, but not by others. The umask is defined by the administrator in the /etc/login.defs file or 022 by default if not specified.

7.1.3.2 Access Control Lists (ACLs)

The TOE provides support for POSIX type ACLs to define a fine grained access control on a user basis. ACLs are supported for all file system objects stored with the following file systems:

- ext3
- tmpfs

An ACL entry contains the following information:

- A tag type that specifies the type of the ACL entry
- A qualifier that specifies an instance of an ACL entry type
- A permission set that specifies the discretionary access rights for processes identified by the tag type and qualifier

An ACL contains exactly one entry of three different tag types (called the "required ACL entries" forming the "minimum ACL"). The standard UNIX file permission bits as described in the previous section are represented by the entries in the minimum ACL.

A default ACL is an additional ACL which may be associated with a directory. This default ACL has no effect on the access to this directory. Instead the default ACL is used to initialize the ACL for any file that is created in this directory. If the new file created is a directory it inherits the default ACL from its parent directory. When an object is created within a directory and the ACL is not defined with the function creating the object, the new object inherits the default ACL of its parent directory as its initial ACL.

7.1.3.3 File system objects

Access to file system objects is generally governed by permission bits. For the ext3 file system, ACLs are supported.

File system objects access checks are performed when the object is initially opened, and are not checked on each subsequent access. Changes to access controls (i.e., revocation) are effective with the next attempt to open the object.

This security function covers FDP_ACC.2, FDP_ACF.1 (1).

7.1.3.4 IPC objects

The TOE implements the following standard types of IPC mechanisms:

- SYSV Shared Memory
- SYSV and POSIX Message Queues
- SYSV Semaphores

Access to the above mentioned IPC mechanisms are governed by UNIX permission bits.

As the IPC objects of UNIX domain socket special files and Named Pipes are represented as file system objects, the access control mechanism covering file system objects are applicable to these IPC mechanisms too.

The TOE maintains IPC object types where each process has its own namespace for that object type: sockets - including network sockets. Access to the socket is only possible by the process whose socket namespace contains the socket reference. Setting of permissions for such objects can be handled using file descriptor passing.

This security function covers FDP_ACC.2, FDP_ACF.1(2).

7.1.3.5 at and cron jobs queues

cron jobs can only be accessed (read/added/modified/deleted) by the owning user. The TOE maintains cron job queues (i.e. the crontab files) for each user. at job queues are accessible to the root user only. Note that each cron job queue is defined with one crontab file.

The root user can always access every cron job queue.

Access to the cron mechanisms can be limited using the /etc/cron.allow and the /etc/cron.deny files. If the allow file exists, only the users specified in these allow file are allowed to access his at or cron job queue, respectively. In case the allow files do not exist, the deny files are analyzed. Only users specified in the deny files are denied access to his at or cron job queue, respectively.

The at or cron jobs are started with the UIDs/GIDs of the creator of the job.

This security function covers FDP_ACC.2, FDP_ACF.1 (4).

7.1.4 Mandatory Access Control

The TOE supports mandatory access control using sensitivity labels automatically attached to processes and objects. This policy is enforced by the SELinux security module and the TOE specific SELinux policy.

Sensitivity labels consist of a hierarchical part (the level) and a non-hierarchical set of categories.

The SELinux security module attaches a “sensitivity label” as part of the security context to the objects defined in Security Policy specification .

Processes are subjects with associated security contexts. When sending signals using the kill system call, the target process behaves like an object.

In addition a process as a subject also has a security context attached. Each process has an effective or “low” sensitivity label (consisting of a hierarchical level and zero or more categories), and a separate “process clearance” or “high” sensitivity label which must dominate the effective label. The effective level is used for all access checks except for processes with the a specific MLS override attribute. Access control is performed based on the sensitivity labels of the process and the object the process interacts with.

When access attempts by a subject onto an object covered by the Discretionary Access Control are performed, the Mandatory Access Control policy is only enforced after the Discretionary Access Control policy allowed the access attempt. In case the Discretionary Access Control policy denies the access attempt, the denial decision is immediately returned to the calling subject.

Attaching the security context to those objects, evaluating the security context in case of access attempts and managing the security context of subjects and objects is performed by functions that SELinux provides for the kernel hooks defined in the LSM framework. The functions at those hooks ensure that all subjects and objects obtain a security context (including a sensitivity label) when they are created in accordance with the rules of the mandatory access control policy.

To support world-writeable directories or home directories for users which can access the system with different labels, the concept of polyinstantiated directories is implemented by the TOE. Polyinstantiation of directories implies that a user's process can only see the file system objects with the same label that his process is assigned with. Considering the purpose of polyinstantiated directories which tries to separate the file system objects of different labels it is clear that polyinstantiation is not relevant for DAC.

This security function covers all SFRs of FDP_IFC.2, FDP_IFF.1, FIA_ATD.1, FIA_USB.1, FMT_MSA.1(1), FMT_MSA.3(2), FMT_MSA.4(2).

7.1.4.1 at and cron jobs queues

The TOE maintains at and cron job queues for each sensitivity label per user and applies the mandatory access control rules when accessing these queues.

Processes spawned by at or cron are assigned the sensitivity label of the creator of the job.

The at program is not a setuid program; therefore, it cannot be executed by regular users. It would be in violation of the evaluated configuration to change the at program to a setuid program.

7.1.4.2 Export/Import of labeled and unlabeled data

The system supports import and export of unlabeled data. When using single level devices, changes in device level must be performed manually by the administrator and are auditable.

An data archiving tool permits import and export of labeled filesystem data when used by administrators by creating archives that preserve label information.

The TOE IPsec implementation allows assigning labels to network objects and enforcing the mandatory access control policy based on those labels.

The IPsec implementation can be used for encrypted and authenticated network communication which is beyond the scope of this Security Target. IPsec is only supported for the purpose of labeled networking, and only in transport mode. Tunnel mode is not supported.

This security function covers all SFRs of FDP_ETC.1, FDP_ETC.2, FDP_ITC.1, FDP_ITC.2, FPT_TDC.1.

7.1.5 Cryptographic services

The TOE provides the NSS library which is covered by a FIPS 140-2 certificate - other cryptographic services implement the cryptographic mechanisms as asserted by the vendor. The NSS library is used in a FIPS 140-2 compliant mode by the following services:

- Wrapper application - providing users with access to the general-purpose cryptographic services
- TSF integrity check - using the cryptographic services of the NSS library to implement the TSF integrity verification mechanism

7.1.5.1 NSS wrapper application

In the evaluated configuration, any user requesting general services from the NSS library shall only use the provided wrapper application to interact with the cryptographic mechanisms of the NSS library. User applications must not link with the NSS library directly as the proper operation of these services for the caller cannot be guaranteed.

The NSS wrapper allows provides the following services to any caller:

- Generation of symmetric and ECDSA keys
- Destruction of symmetric and ECDSA keys
- Encryption and decryption using the AES cipher
- Signature generation using the ECDSA mechanism
- Message digest generation using the SHA-2 family

The NSS library uses a deterministic random bit generator seeded by /dev/urandom. This file provides access to the kernel-maintained non-blocking entropy pool which is filled based on first and second derivation of the time deltas between the occurrence of selected hardware interrupts. The kernel uses carefully selected hardware interrupt sources to prevent attackers from predicting the entropy. In case the entropy pool runs low on entropy, the kernel applies a deterministic random number generation mechanism utilizing the SHA-1 algorithm until sufficient entropy can be obtained from the interrupt sources.

This security function covers all SFRs of FCS_BCM_EXT.1, FCS_COP.1(1), FCS_COP.1(2), FCS_COP.1(3), FCS_RBG_EXT.1, FCS_CKM.1(1), FCS_CKM.1(2), FCS_CKM.4, FCS_COA_EXT.1.

7.1.5.2 TSF integrity check

The TOE implements an integrity verification tool which maintains a database with SHA-256 hashes of all TSF binary files as well as the meta data of these files (such as permission bits, owner ship information, modification time, file system object name).

The integrity verification mechanism scans the TSF binary files and other configured files and matches each file with the attributes stored in the database. If an attribute does not match, it generates a warning.

Besides the verification of the integrity of the TSF binary files, the integrity verification mechanism can also be used to update the integrity check database in case file system objects under control of that mechanism are intentionally changed. The database is accessible by root only and can therefore only be updated by an authorized administrator.

To calculate the hash values of files, the integrity verification mechanism uses the NSS library services.

This security function covers all SFRs of FPT_TST_EXT.1.

7.1.6 Security Management

The security management facilities provided by the TOE are usable by authorized users and/or authorized administrators to modify the configuration of TSF. The configuration of TSF are hosted in the following locations:

- Configuration files (or TSF databases)
- Data structures maintained by the kernel and within the kernel memory

The TOE provides applications to authorized users as well as authorized administrators to perform various administrative tasks. These applications are documented as part of the administrator and user guidance. These applications are either used to modify configuration files or to access parameters controlled and enforced by the kernel via kernel-provided interfaces to user space.

Configuration options are stored in different configuration files. These files are protected using the DAC mechanisms against unauthorized access (note that although these files are also covered with MAC protection, that protection is considered to be irrelevant for ensuring information flow control as only the administrator is able to access them to add unspecified information). It is the task of the persons responsible for setting up and administrating the system to ensure that the access control features of the TOE are used throughout the lifetime of the system to protect those databases. These configuration files are accessed using applications which are able to interpret the contents of these configuration files. Each TOE instance maintains its own TSF database. Synchronizing those databases is not performed in the evaluated configuration. If such synchronization is required by an organization it is the responsibility of an administrative user of the TOE to achieve this either manually or with some automated assistance.

To access data structures maintained by the kernel, applications use the kernel-provided interfaces, such as system calls, virtual file systems, netlink sockets, and device files. These kernel interfaces are restricted to authorized administrators or authorized users, if applicable, by either using DAC (for virtual file system objects) or special kernel-internal verification checks for each interface.

The TOE provides security management applications for all security-relevant settings listed throughout this ST, i.e. all FMT_MSA.1 and FMT_MTD.1 iterations.

This security function covers all SFRs mapped to FMT_SMR.1, FMT_MOF.1(1), FMT_MSA.1(1), FMT_MSA.1(2), FMT_MSA.3(1), all iterations of FMT_MTD.1, FMT_REV.1(1), FMT_REV.1(2), FMT_SAE.1, FIA_SOS.1, FIA_UAU.7, FMT_MSA.2, FDP_ACC.2, FDP_ACF.1(1), FDP_RIP.2, FDP_RIP.3.

7.1.7 TSF Protection

While in operation, the kernel software and data are protected by the hardware memory protection mechanisms described in the high level design and the hardware reference manuals for the underlying hardware. The memory and process management components of the kernel ensure a user process cannot access kernel storage or storage belonging to other processes.

Non-kernel TSF software and data are protected by DAC and process isolation mechanisms. In the evaluated configuration, DAC permission settings ensure that files that are part of the TSF database as well as files and directories containing internal TSF data (e.g. batch job queues) are also protected from unauthorized modification and reading.

The TSF including the hardware and firmware components are required to be physically protected from unauthorized access. The kernel mediates all access to the hardware mechanisms, other than program visible CPU instruction functions and main storage defined by the kernel to be directly accessible by a user process.

The boot image for each host with the evaluated TOE is adequately protected using proper DAC permission settings.

7.1.7.1 TSF Invocation Guarantee

All system protected resources are managed by the TSF. Because all TSF data and the associated TSF data structures are protected, these resources can be directly manipulated only by the TSF using defined TSF interfaces. This satisfies the condition that the TSF must be "always invoked" to manipulate protected resources.

Resources managed by the kernel software can only be manipulated while running in kernel mode.

Processes run in user mode and can call functions of the kernel only as the result of an exception or interrupt. The hardware and the kernel software handling these events and ensure that the kernel is entered only at pre-determined locations, and within pre-determined parameters. All kernel managed resources are protected such that only the kernel software is able to manipulate them.

Trusted processes implement resources managed outside the kernel. The trusted processes and the data defining the resources are protected as described above depending on the type of interface. For directly invoked trusted processes the program invocation mechanism ensures that the trusted process always starts in a protected environment at a predetermined point. Other trusted process interfaces are started during system initialization and use well defined protocol or file system mechanisms to receive requests.

Some system calls or parameter of system calls are reserved are reserved for trusted processes. When called the kernel checks that the calling process runs with an effective userid of 0.

7.1.7.2 Kernel

The TOE software consists of a privileged kernel and a variety of non-kernel components (trusted processes). The kernel operates on behalf of all processes (subjects).

The kernel runs in the CPU’s privileged mode and has access to all system memory. All kernel software, including kernel extensions and kernel processes, execute with kernel privileges and are part of the TSF. The kernel is entered by some event that causes a context switch such as a system call, I/O interrupt, or a program exception condition.

Upon entry the kernel determines the function to be performed, performs it, and, when finished, performs another context switch to return to user processing (eventually on behalf of a different subject).

The kernel is shared by all processes, and manages system wide shared resources. It presents the primary programming interface for the TOE in the form of system calls.

Because the kernel is shared among all processes, any process running "in the kernel" (that is, running in privileged hardware state as the result of a context switch) is able to directly reference the data structures that implement shared resources.

The major components of the kernel are memory management, process management, the file system, the system call interface, and the device drivers.

The TOE supports dynamically loadable kernel modules that are loaded automatically on demand. Kernel modules are actually a part of the kernel that is not resident but loaded as part of the kernel when needed. Whenever a program wants the kernel to use a feature that is only available as a loadable module, and if the kernel hasn't got the module installed yet, the kernel will invoke a user space application which looks for the requested module and loads it using system calls.

7.1.7.3 Trusted Processes

Trusted processes in the TOE are processes running in user mode but with root privileges.

A trusted process is distinguished from other user processes by the ability to affect the security policy. Some trusted processes implement security policies directly (e.g., identification and authentication) but many are trusted simply because they operate in an environment that confers the ability to access TSF data (e.g., programs run by administrative users or during system initialization).

The major functions implemented with trusted processes include user login (identification and authentication), batch processing, some network operations, system initialization, and system administration.

The kernel will check for each system call that requires root privileges if the process that issued the call has those privileges. If not, the kernel will refuse to perform the system call. The kernel will also check for each access to an object protected by the any of DAC mechanism, if the process has the required access rights for the attempted type of access.

Any program executed with root privileges has the ability to perform the actions of a trusted process. It is therefore important that a site operating the TOE system strictly controls those programs and prohibits that those programs are modified or that programs from untrusted sources are executed with root privileges.

Trusted processes are part of the TSF.

7.1.7.4 Secure failure state

The system provides a single user maintenance mode. The system can be configured to automatically enter single user mode when the self test utility detects a security failure. The self test is performed during boot time.

In single user mode, all interactive user sessions are terminated and all system daemons that can run tasks on a user's behalf (crond) are unavailable.

An authorized system administrator can use the system console to interact with the system and re-enter normal multiuser mode.

7.1.7.5 Resource limits

The TOE controls the usage of resources by the subjects. Resource limits can be configured by authorized administrators and are enforced by the TSF.

The following resource limits are provided:

- Session limit - The number of concurrent sessions of one user can be limited to an administrator configurable number. The TOE enforces this limit using a PAM module.
- File system quota - The amount of file system storage space usable by one user identified with his user ID can be limited to an administrator configurable limit. That limit is enforced by the kernel.

This security function covers the SFRs of FPT_RCV.1, FRU_RSA.1, FTA_MCS.1.

As the TOE is only executed on one hardware system and does not rely on other systems for enforcing the security functionality, the TOE is considered to be not a distributed system. In that effect, FPT_TRC_EXT.1, FPT_ITT.1, FPT_ITT.3 are not applicable and therefore trivially met by the system architecture.

8 Abbreviations, Terminology and References

8.1 Abbreviations

ACL

Access Control List

API

Application Programming Interface

HTTP

Hypertext Transfer Protocol

SFR

Security Functional Requirement

SSL

Secure Sockets Layer

ST

Security Target

TCP/IP

Transmission Control Protocol / Internet Protocol

TLS

Transport Layer Security

TOE

Target of Evaluation

TSF

TOE Security Functionality

VM

Virtual Machine

VPN

Virtual Private Network

8.2 Terminology

This section contains definitions of technical terms that are used with a meaning specific to this document. Terms defined in the [CC] are not reiterated here, unless stated otherwise.

Authentication Data

This includes the password for each user of the product. Authentication mechanisms using other authentication data are not supported in the evaluated configuration.

Authorized Administrator

This term refers to a user in one of the defined administrative roles of a Linux system. The TOE associates the user with the UID of zero and named "root" with administrative authorities. Effectively, the UID zero is assigned with all Linux capabilities known to the Linux kernel. Every user who is allowed to log on as that root user or to switch their UID to the root user is considered an authorized administrator. In addition, any user who is able to execute applications which grant one or more Linux capabilities to be used in an unconditional manner is considered an authorized administrator. Note: the process executing on behalf of the root user must possess MLS override attributes to perform management aspects of the Mandatory Access Control Policy.

Classification

A sensitivity label associated with an object.

Clearance

A sensitivity label associated with a subject or user.

Data

Arbitrary bit sequences on persistent or transient storage media.

Dominate

Sensitivity label A dominates sensitivity label B if the hierarchical level of A is greater than or equal to the hierarchical level of B, and the category set of label A is a proper subset of or equal to the category set of label B. (cf. Incomparable sensitivity labels).

Information

Any data held within a server, including data in transit between systems.

Named Object

In Linux, those objects that are covered by access control policies. The list of objects defined as named objects is provided with FDP_ACC.1.

Object

For Linux, objects are defined by FDP_ACC.1.

Product

The term product is used to define software components that comprise the Wind River Linux system.

Sensitivity Label

The TOE attaches a sensitivity label to each named object. This label consists of a hierarchical sensitivity level and a set of zero or more categories. The policy defines the number and names of the sensitivity levels and categories.

Subject

There are two classes of subjects in WRLS: i) untrusted internal subject - this is a Linux process running on behalf of some user or providing an arbitrary service, running outside of the TSF (for example, with no privileges); ii) trusted internal subject - this is a Linux process running as part of the TSF (for example: service daemons and the process implementing the identification and authentication of users).

Target Of Evaluation (TOE)

The TOE is defined as the Wind River Linux operating system, running and tested on the hardware and firmware specified in this Security Target. The BootPROM firmware as well as the hardware form part of the TOE as required by the NIAP interpretation for a TOE that relies on hardware / firmware functions to implement this proper separation and isolation mechanisms required by ADV_ARC.1.

User

Any individual/person or technical entity (such as a service added by the administrator on top of the TOE) who has a unique user identifier and who interacts with the Wind River Linux product.

User Security Attributes

Defined by functional requirement FIA_ATD.1, every user is associated with a number of security attributes which allow the TOE to enforce its security functions on this user. This also includes the user clearance which defines the maximum sensitivity label a user can have access to.

8.3 References

CC **Common Criteria for Information Technology Security Evaluation**
Version 3.1R3
Date July 2009
Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R3.pdf
Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R3.pdf
Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R3.pdf

ECG **EAL4 Evaluated Configuration Guide for WindRiver Linux Secure 1.0**
Version 1.0
Date 2011-01-14
File name WRLS1-EAL4-Configuration-Guide.pdf

niap-ospp **US Government Protection Profile for General-Purpose Operating Systems in a Networked environment**
Version 1.0
Date 2010-08-30

WRLSAG **Wind River Linux Secure Administrator's Guide**
Version 1.0
Date 2011-01-14
File name wr_linux_secure_admin_guide_1.0.pdf

WRLSCG **Wind River Linux Secure Configuration Guide**
Version 1.0
Date 2011-01-14
File name wr_linux_secure_config_guide_1.0.pdf

