TABLE OF CONTENTS

About This Document...

If you are new to Trustifier, you should start with *Trustifier Security Model and Concepts Guide*.

You are encouraged to peruse the reference guide to familiarize yourself with the information available.

New versions, addenda and errata of this manual are available at http://www.trustifier.com/manual

About Trustifier Security Administrator's Guide

The Trustifier Security Administrator's Guide introduces Trustifier Security to a security administrator, and provides pointers to further documentation. This section does not aim to be complete. In places within this section, precision in certain areas has been omitted to get the general concepts across. To get precise information about Trustifier internals also refer to the *Trustifier Reference* section.

About Trustifier Reference

The *Trustifier Reference* is a complete reference manual to Trustifier's concepts, internals, API and usage.

Conventions

Throughout this document several typographic and language conventions are used to facilitate readability.

Typographic Conventions

The following typographic conventions are used throughout this document:

Convention	Description
mono-space bold	Literal. Type these words exactly as you see them
< >	Token. Items contained within angle brackets represents a concept, and is to be replaced with appropriate item represented thereby. For example, <*number*> should be replaced with a number.
[ ]	Optional token or literal. The items contained in square brackets is to be typed without the brackets, and is optional. For example, [**on**] indicates that the literal word **on** could be typed, and [<*options*>] indicates that optional parameters could be typed.

Convention	Description
|	Alternative(s). The vertical bar separates alternatives between tokens and literals. For example **on**|**off** indicates that either **on** or **off** could be typed.
…	You can optionally repeat indicated option.

Iconic Conventions

The document uses several icons to indicate important points

Icon	Description
	Cautionary Note – Read this note carefully as it indicates an operation you should not perform under normal circumstances.
!!	Cautionary Note – Indicates privilege incapacitating operational modes, typically used for very high security environments.

Language Conventions and Definitions of Terms

The words and language convention used is as below.

Chapter Headings

The headings of chapters follow the UN*X manual page conventions, indicated chapter headings contain the following:

Synopsis	Command synopsis. This section summarizes the commands that you can use to control Trustifier and execute indicate operations.
Description	Describes, in detail, each of the items in the synopsis.
Examples	Provides usage examples
See Also	Provides cross references to other (related documentation)

Any other chapter headings are topic indicative.

Definition of Terms

Throughout the document the terms mean the following:

Trustifier	Refers to the entire security system suite

trustifier Trustifier	Refer to the front-end manager program for
tkm	Refers to the Trustifier kernel module

TRUSTIFIER

SECURITY MODEL AND CONCEPTS GUIDE

INTRODUCTION

This section provides an overview of the Trustifier Security Model. If you are new to Trustifier, this is the place to start.

Trustifier's security model is a different – and we feel a more intuitive and easier – approach to Multi-Level Secure, Roles Based Access Control (RoBAC), and Rules-Based Access Control (RuBAC).

This guide provides an overview of how Trustifier works and helps to familiarize you to Trustifier semantics.

SOME DEFINITIONS

Objects, Subjects and Owners

Object

An Object is any data input or data output resource in the operating system. It could be a file on a disk, a network socket, a storage device, a display or a printer.

Subject

A Subject is the process that is manipulating the Objects.

Owner

An Owner is the user or the group that owns an Object or a Subject. An Owner always has a defined user-id and group-id on the system. Typically owners have accounts on the system, but this is not always necessary.

OWNER-CENTRIC SECURITY

Trustifier is an Owner-Centric (as opposed to Object or Subject centric) security system. Owner-Centric security means that security rules are defined in terms of the Owners of Objects and Subjects, not just the objects and subjects of manipulation.

Trustifier security rules are intuitive to the way we think in organizations.

Consider the following example, lets say Jill has to have reading access to Bob's engineering files with a sensitivity of secret or less for work.

Owner Centric Rule

Owner centric rule would be 'Bob trusts Jill with files of secret sensitivity in engineering group'.

The primary nature of this rule is that it is atomic in description and operation.

Object/Subject-Centric Rule

On the other hand, an object/subject centric rule for the same could be:

'For all files that are owned by Bob in Engineering or will be created by Bob in the Engineering group where the sensitivity is secret or less, create an Access Control List entry for Jill on the file for Read access'.

A rather convoluted rule, which is as convoluted to implement as it is to understand. This rule, is not atomic and really has to be broken down into several steps before it can be implemented or enforced.

More steps means less reliability, since each step has to be complete. Also, there could be hidden security implications during state change in multiple steps. Lastly, multiple steps are harder to verify for accurate compliance with the business rules.

Trustifier Security Rules and Settings

Trustifier defines several levels of security rules that work together to create an information-manipulation or information-release path.

In all cases the initiator

Trustifier security settings may be classified into the following sub sections:

1. Service Access Control (Rules-based Access Control)

   Controls access to services and I/O through rules

2. Privilege Grants (Rules-based Access Control)

   Assigns privilege through rules

3. Relative Trust and Ranks (Roles-based Access Control, MLS and Domain Separation)

   Assign privileges through roles and trusts

4. Auditing (Rules and Roles Based Auditing)

   Monitoring and logging security related events.

5. Contextual and Parametric Determination

   Allows enforcement of any RuBAC and RoBAC under specific contexts

Each of the above is described in the sections that follow.

Understanding Service Access Control (RuBAC)

Trustifier assigns Services Access Control is a rules-based access control subsystem. At the core is a straightforward per Owner Allow/Deny map for each system services request and I/O request.

This

Using the **Limits Module** the user can assign com

Control system calls access

Trustifier allows you to set what system calls a user is allowed to make. By simply flicking a bit, a user can be brought under severe audit and control.

Applications

The concept of allowing the bare minimum of what's needed is very powerful on its own and doesn't require lengthy explanations. However, here are some simple examples that outline the underlying principle:

Remove privilege escalation system calls like setuid, except for the users who really need it. This greatly limits the holes in security and provides for a tighter security profile.

Remove exploitable system calls (e.g. ptrace), and only give it to users who need the functionality.

Remove access to system information (e.g. syslog) calls that give potentially exploitable information to users.

Control I/O access

Set which IOCTL, SOCKET and FCNTL operations a user is allowed to request. This is similar to limiting system calls; however it acts on different set of capabilities. Using Trustifier, we can limit the user's I/O control to specific operations on the input and output channels.

Applications

Applications of Limiting I/O control are slightly different from limiting system calls. Interesting and effective control can be achieved relatively easily by switching on or off certain access. For instance we can limit:

- Network protocols and protocol features that a user can access (e.g. ability to use GRE over IP, or the permission to use RAW sockets).
- The devices a user can access or manipulate (e.g. discs).
- Memory segments (Inter-Process Communication – IPC) that a user can access or manipulate.

UNDERSTANDING PRIVILEGE GRANTS

Role-based privileges set which administrative capabilities (as defined within the O/S kernel) a user may be granted.

Roles based privileges allows the dispersion of administrative capabilities among several users. Privileges traditionally associated with super user can be independently enabled and disabled for non-root and root users, giving a more controlled environment.

Applications

Using Role-based privileges it is possible to grant users specific roles within the system. For examples:

Daemon binding: Some protocols use privileged ports to which only the super-user (root) is able to bind. With Role-based Privileges it is possible to grant a non-root user the ability to bind to a privileged port.

Multicasting: An otherwise unprivileged user can be granted the ability to use multicasting network protocols.

Reserved File System Space Access: Create an auditing manager that has the capability to use reserved file system space.

File Management: Enable a specific user to have the capabilities to change ownership and permissions including setuid and setgid.

Network management: Create a network administrator that is just allowed to manage the system's network devices.

Trustifier supports all of POSIX.1e role-based privileges and introduces some of its own. Using role-based privileges it is possible to grant a user relatively narrow scope of privileged operations, safeguarding the system from harm.

UNDERSTANDING RANKS

Trustifier implements a relative ordering or Ranking of Owners to provide Roles Based Access Control, Mandatory Access Control, Multilevel Security and Domain Separation.

Definition of Rank

The Rank of an Owner is a number between 1 and 127 assigned to the Owner in relation to another Owner in a Rank Class.

Rank Classes

There are essentially two classes of ranks:

1. Secrecy
2. Integrity

Thus an Owner can have a secrecy ranking or an integrity ranking with respect to another Owner in context of the binding Owner.

When creating mandatory access control, two types of relationships are of interest. One is the traditional **secrecy** relationship the other is **integrity** relationship.

Secrecy is the traditional mandatory access control piece, where the higher user has read-access to the files of lower-ranking user; however the lower-ranking user does not have access to higher ranking user's data. In addition, under the secrecy relationship, higher ranking users cannot create data that ranks lower than their rank. This relationship is called write-up read-down in security texts.

Integrity ranking is the reverse of secrecy. Here data is normally accessible for reading and is shared, but by making file ownership immutable and unchangeable, a signature on the content is maintained. Thus the "integrity of data" is determined by the ownership of the file.

Ranking relationships

The following table summarizes the level of access, in terms of **read (R), write (W), execute (X) and create (C)** operations, that may be performed by a user on an object (mostly files and directories) when a subject⇔object relative trust ranking exists.

<table>
<tr><td colspan="2"></td><td colspan="5">SUBJECT RANK (R_S)</td></tr>
<tr><td colspan="2"></td><td>NO RANKING</td><td colspan="2">SECRECY RANKING</td><td colspan="2">INTEGRITY RANKING</td></tr>
<tr><td rowspan="6">OBJECT RANK (R_O)</td><td>NO RANK</td><td>R,W,X,C</td><td colspan="2">R,X</td><td colspan="2">R,X</td></tr>
<tr><td rowspan="4">SECRECY RANKING</td><td rowspan="4">NO ACCESS</td><td>RANK COMPARISON</td><td>ALLOWED OPERATIONS</td><td colspan="2" rowspan="4">NO ACCESS</td></tr>
<tr><td>$R_U < R_O$</td><td>NO ACCESS</td></tr>
<tr><td>$R_U = R_O$</td><td>R,W,X,C[1]</td></tr>
<tr><td>$R_U > R_O$</td><td>R,X</td></tr>
<tr><td rowspan="4">INTEGRITY RANKING</td><td rowspan="4">R,X</td><td colspan="2" rowspan="4">R,X</td><td>RANK COMPARISON</td><td>ALLOWED OPERATION</td></tr>
<tr><td>$R_U < R_O$</td><td>R,X</td></tr>
<tr><td>$R_U = R_O$</td><td>R,W,X,C</td></tr>
<tr><td>$R_U > R_O$</td><td>R,W,X</td></tr>
</table>

[1] Creation is only allowed inside secret containers

Rank Variants

There are four Rank variants under normal Linux where each object has one user and one group owner:

$R(u_s, u_o)$ User to User Rank.

R(u, g) User to Group Rank

R(g, u) Group to User Rank

$R(g_1, g_2)$ Group to Group Rank

User to User Rank

R(u, u) or the Rank of a user in relation to another user is a relative value of how much one user trusts another.

Formally,

If User u_o trusts u_s with secrets of Rank *r* or less, then u_s can access u_o object where u_o has a ranking of *r* or less in secrecy with respect to the group Owner of the object.

For example,

If a user bill has a file my_design in group engineering, and bill's secrecy rank is 3 then ted can access my_design if R(bill, ted) >= 3. Other conditions may apply before flow of information may be allowed.

The relationship is asymmetric, that is to say:

$$R(u_1, u_2) \not\Box R(u_2, u_1)$$

User-User ranks allow lateral flow of information across, otherwise, separated domains.

User to Group Rank

Rank of a user in relation to a group indicates the clearance of the user within the group.

The Subject Owner may access any Object within the Object Owner Group if Subject Owner's Rank in the Object Owner's Group is greater than or equal to the Rank of the Object Owner's User in the Object Owner's Group.

MLS and Domain Separation

When used with secrecy class the User to Group Rank is essentially Multilevel Security with Domain Separation.

Group to User Rank

Rank of a Group in relation to a user. This is essentially how much a particular user trusts an entire group with their sensitive information. The implications are

similar to User to User ranks, however the decision applies to any one acting with the appropriate effective group id and does NOT have a lower user-user trust ranking.

Group to Group Rank

Group to Group ranking determines how much information can flow from one group to another group.

Any ranked member of the (Subject) group that in-turn has a ranking in the (object) group can access information of the object group up to their rank in the subject group or the rank of the subject group in the object group, whichever is lower.

Example

Say that the following conditions exist:

1. bob has a secrecy ranking of 3s in manufacturing group,
2. rebecca has a secrecy ranking of 5s in manufacturing group
3. john has a secrecy ranking of 2s in engineering group,
4. ted has a secrecy ranking of 6s in engineering group,
5. jill has a secrecy ranking of 5s in the research group,
6. manufacturing group has a secrecy ranking of 4s in the engineering group.
7. engineering group has a secrecy ranking of 5s in manufacturing group.
8. research has a secrecy ranking of 6s in engineering and 6s in manufacturing groups.

Then,

Bob can access up to 3s secrets in manufacturing group and only create 3s secrets.

Rebecca can access up to 5s secrets in manufacturing group and only create 5s secrets.

John is only able to access secrets of up to 2s in engineering group, and is unable to see anything belonging to Bob or Rebecca even though engineering group outranks them in manufacturing group.

Jill can access everything belonging to Bob, Rebecca and John in the engineering and manufacturing groups. She cannot access Ted's documents. She can create 5s files in research group.

Ted can access Bob, Rebecca and John's files in manufacturing and engineering groups respectively and create 6s secrets in engineering group.

Secrets container

There are caveats. Files with secrecy ranking can only be created inside a directory that belongs to the same group, called the secrets container. Files cannot be ranked higher than their container. A container's secrecy ranking is the same as that of the owner within the group that owns the directory.

The following summarizes access relationship with a Secrets Container:

User Secrecy Ranking > Secrecy Ranking of the Container

The user may list all files in the container. The user may not create new files.

User Secrecy Ranking == Secrecy Ranking of the Container

The user may list all files in the Container and create new files.

User Secrecy Ranking < Secrecy Ranking of the Container

The user may list files she owns, but she will not be able to list files owned by other users[2]. The user may create new files if normal directory permissions allow such creation.

What about normal file permissions?

While, secrecy and integrity rankings impose further limitations on access, they do not replace them. In other words the ranking augments the standard file permissions and the user is, even after being allowed access as per rankings, still subject to normal system file permissions.

Why group-wise relative ranking?

The main case for creating the concept of relative ranks in groups rather than using something like Bell-La Padula (security) or BIBA (integrity) labeling systems is the integration with concepts and approaches that are already understood by system security officers and system administrators:

The first and most important advantage is that the concept of users belonging to group(s) is familiar. Now we simply introduce the easy concept of relative positioning of a user in a group – which, incidentally, is how most, if not all, organizations are organized – we've managed to map our security access control to organizational structure very intuitively. For instance, if Betty is Bob's superior in a project, just give Betty a higher ranking than Bob in the project group.

The second advantage is the trivial case of no ranks. The group-wise relative ranking collapses to normal permissions when no ranks are specified making

2 A user may be able to deduce the existence of a file or directory name by attempting to create the file or directory of that name. If normal Unix File Permissions should allow the creation of the file and permission is denied when attempting to create said object the user can assume the existence of an object of that name with a high degree of certainty. Any systematic attempt to exploit this is easily discerned by auditing.

transition into mandatory access control very manageable and affordable, in time, effort and resources spent.

Applications

Secrecy

Any "need-to-know" compartmentalization of information can be enforced by use of secrecy ranks. From National Defence applications, to protection of intellectual property, to managing access to corporate sensitive information. The applications are innumerable. However, let's look at some of the more interesting and esoteric examples:

Caging daemons

By giving a suspect daemon process, e.g. a web server, a **high** secrecy ranking in a benign group makes it impossible for an attacker to exploit the vulnerabilities in the process and create, modify, or remove files from the normal system.

Controlling input/output devices

By giving a ranking to devices like printers, screens, keyboards, even network cards, we can limit the input or output of sensitive information to and from devices.

RAM and SWAP

By simply giving kernel's memory interface and swap-space a higher rank, we can readily block out raw ram and swap scans.

Integrity

Examples where integrity becomes useful are a bit more difficult to readily perceive, however Integrity ranking is a very powerful tool as will be evident after you read the following examples:

System and Administration Separation

By simply ranking the operating system files higher than the system administrator, the core system becomes read-only to all users. Disallowing simple mistakes like: **rm –rf /**

Ranking devices

By giving block devices a ranking, we can ensure that critical devices are immutable by system administrator. For instance, if a block device has a higher rank than system a system administrator, she cannot change the partition tables of a disc without proper authorization from the security officer.

Contextual and Parametric Determinations

Please Note these features are only available when Limits Module are installed.

Trustifier supports contextual and parametric determination of enforcement rules. Using the **Limits Module** contextual and parametric controls allow rules that examine current state of the Owner, the Subject and the Object and any historic control flags to determine whether or not to enforce a control and audit decision.

It is best to show both the concepts and Trustifier solutions as examples:

Example 1

Define the rule:

If Bill has a socket open do not allow Bill to open a file with sensitivity of secrecy 3.

Solution:

trustifier deny open to bill when subject:tcp-sockets.count>0 and object:rank.value>=3 and object:rank.type=s

Example 2

Define the rule:

If Bill has previously accessed objects of value 5s or more do not allow the user to access to USB storage devices

Solution:

First let's remember it if bill ever access a sensitive object (creating a historic flag).

trustifier audit open for bill flagging bill-is-marked when object:rank.type=s and object:rank.value>=5

Now if bill has accessed a sensitive object in the past (identified by bill-is-marked flag) and bill tries to open an object where the object's target is a USB device then deny the open.

trustifier deny open to bill when bill-is-marked and object:device.bus.name=USB

For more information on all the predefined and definable objects see *Limits Module Reference.*

TRUSTIFIER

COMMAND-LINE REFERENCE

TRUSTIFIER COMMAND-LINE REFERENCE

SYNOPSIS

Trustifier command line tool may be used in the following ways:

trustifier [**-P** <*pass-phrase*>] [<*module*>**:**]<*command*> [<*options*>]

Perform an action, where *command* is any command installed by Trustifier modules. See the complete list in Table 1 (on the next page).

trustifier --help

List help

trustifier -l|--list-commands

List all of the commands available

trustifier -L|--list-modules

List all of the installed modules

Upon execution trustifier may interactively ask for a pass-phrase for security sensitive tasks.

The table below provides a complete list of commands available. Trustifier version information indicates the Trustifier release that supports the indicated command.

Table 1 - Trustifier Commands

Command	Module	Description	Versions
allow	user	Allow a system call to a user, group or system	*All*
apply	user	Apply an existing user as a template to a user, group or system	*All*
audit	user	Set audit flags for a user, group or system	*All*
deny	user	Deny a system call to a user, group or system	*All*
grant	user	Give special privileges to a user, group or system	*All*
list	base	Lists system calls, access control flags, and user trust flags on the current environment	*All*
load	db	Load persistent database into the kernel	*All*
mode	base	Shows current mode of operations	*All*
noaudit	user	A synonym for unaudit	*All*
password	base	Change Trustifier security administrator's password	*All*
lock	base	Lock ALL changes to Trustifier	*All*
rank	user	Set a multi-level secure ranking of a user, group or system	*All*
register	user	Create a separate security table in the kernel for a user or a group	*All*
restore	db	Synonym for **load**	*All*
revoke	db	Revoke a granted privilege	*All*
save	db	Save kernel database to persistent storage	*All*
set	base	Set current mode of	*All*

Command	Module	Description	Versions
		operation	
show	user	Show the current security table for a user, group or system	*All*
start	base	Start Trustifier operations	*All*
status	base	Show current Trustifier status	*All*
stop	base	Stop Trustifier operations	*All*
ssn	*	Secure safety number	*All*
unaudit	user	Clear audit flags for a user, group or system	*All*
unregister	user	Un-register a user or group (dropping them to System-Wide default least privilege)	*All*
verify	db	Verifies the consistency of the persistent database	*All*

DESCRIPTION

trustifier is the front end for managing and controlling the Trustifier™ Advanced Security System for Linux.

trustifier requires back-end modules to provide the actual management functionality and is fully extensible.

Depending upon the Trustifier modules installed on the system, **trustifier** provides the various commands that you may be able to invoke.

Listing all commands

To see all the commands available by the currently installed modules simply type:

trustifier -l

This will list all of the commands available to you from each module:

base : list mode passwd password permlock set showpass start status stop

db : load restore save ssn verify

user : allow audit deny grant noaudit rank register revoke show unaudit unregister

⋮

Of course your list may be different.

Commanding Trustifier

To give Trustifier an instruction simply type:

trustifier <*command*>

Where *command* is one of the installed commands. For example:

trustifier start

If two different modules provide the same command you have to specify the module, as in:

trustifier <*module*>**:**<*command*>

For example, both db and ui modules provide the ssn command. So, if you want to invoke the ssn command for the db module you would type:

trustifier db:ssn

Getting help at the command-line

All trustifier commands support the **--help** flag that provide the pertinent content of this manual. When in doubt you can always ask for help using:

trustifier <*command*> **--help**

For example,

trustifier set --help

SECURITY OF OPERATIONS

Trustifier requires internal authentication for any changes that will affect the security. It also requires authentication to release any potentially sensitive data about the current state of security.

Last line of defense

Having an internal kernel-based authentication mechanism ensures that the system is not abused by the system administrators who do not have security administration rights: Namely system administrators may be able to acquire the user-id of a particular security administrator through clandestine means, but will still have a last line of defense to pass through before system breach.

Specifying Pass-phrase on command-line or in scripts

While **trustifier** allows the security officer to specify the pass-phrase on the command line we vehemently discourage you from doing so. Specifying pass-phrase on command-line or in scripts a very bad idea from security standpoint, and most normal operations including the boot-up process do not require this!

TRUSTIFIER:BASE – TRUSTIFIER BASE MODULE

Base operations module for Trustifier Advanced Security System. The commands in the base module control the over-all behavior of the Trustifier kernel security enforcers.

SYNOPSIS

trustifier start|**stop**

Start or stop system-wide Trustifier protection.

trustifier mode [**-v**|*<mode>*]

Display current operation mode [verbosely] or set the mode.

trustifier status

Synonym for **mode -v**

trustifier set [**-h**|**--help**]

Show help on operational flags that can be set.

trustifier set *<ops-flag>* **on**|**off** [*<ops-flag> on*|*off...*]

Set modes of operation.

Where *<ops-flag>* is any of:

[**extended**] **control**

Enforce [extended] mandatory access control on system service requests.

privilege grants

Respect privilege grants and user-trust flags on service requests to non-root users.

success|**failure**|**privilege audits**

Audits success, failure and privilege grants on system service requests.

c2 audits

Produce Orange-Book C2 format audit trails.

secrecy

Enforce write-up read-down multi-level security on ordered groups.

integrity

Enforce write-down read-any multi-level data non-repudiation and integrity on ordered groups.

strict groups

User effective group id must be explicitly set to the ordered group for MLS privilege escalation for integrity or secrecy enforcement.

real-uid maps

Use real user-id to lookup access and privilege control. Trustifier uses the effective user-id otherwise.

de-escalation

Set or clear de-escalation locking.

trustifier password|passwd

Set security administrator pass phrase.

trustifier list [**--index**] <*control-category*> [<*control-category*>...]

Lists system's kernel control flags (with kernel indices).

Where <*control-category*> is any of the following:

call

Kernel system calls.

lcap

POSIX.1e (or similar) kernel administrative privilege capabilities.

utfl

User trust flags and supported Trustifier-specific capabilities.

trustifier permlock

Lock settings in place permanently. CAUTION !! this is a non reversible operation and incapacitates system completely. privilege escalation

DESCRIPTION

The **trustifier:base** module provides the core commands to manipulate the Trustifier Advanced Security System.

Starting and Stopping Trustifier

To start Trustifier Advanced Security System type:

trustifier start

This will signal the Trustifier Kernel Module to activate protection according to the current control and audit mode.

To stop Trustifier Advanced Security System type:

trustifier stop

NOTE: that the stop command does not remove Trustifier Kernel Module from memory, nor does it loose any of the specific settings, it simply turns the control and auditing off.

Viewing and changing Trustifier mode of operation

Viewing current mode

The Trustifier mode of operation tell what control and auditing features of Trustifier are currently active. The mode is represented by an octal number. To view the current mode of operation type:

```
trustifier mode
```

This will display a five digit octal number, for example

```
trustifier mode
13510
```

Trustifier Mode Octals table below for a complete explanation of the octal flags. The **trustifier set** command can be used to set the mode in a more user-friendly and descriptive manner.

Verbose flag to the mode command gives a descriptive output, for example:

```
trustifier mode -v
Trustifier security is [in-active]
Security Mode (CAS)          :  13351
-- Control Level             :  0003 ([LPAC] Linux CAC with Trustifier MAC)
-- Auditing Level            :  0005 ([AUD*] Audit Failed Accesses and Capability
Grants)
-- UID0 Overrides            :  [Yes]
-- RUID MAC                  :  [No]
-- Extended PMAC             :  [No]
-- C2 logs                   : [Yes]
-- Secrecy                   : [Yes]
-- Integrity                 : [No]
-- Strict Groups             : [Yes]
```

trustifier status is a synonym for **trustifier mode -v**

Setting mode of operation quickly

The quick way of setting Trustifier operations mode is to specify its octal: !!

```
trustifier mode <octal-number>
```

This format of the command also the quickest way to lower the security of the system unintentionally. So to guard against that the Trustifier mode command requires interactive verification before it passes the mode change request to the kernel.

Setting mode of operation safely

The safer way of setting Trustifier operations mode is to specify it using the **set** command.

Enforce System-wide Mandatory Access on System Calls

Enforce core and extended mandatory access control on system-calls by turning the system call control using:

trustifier set [**extended**] **control on**

Turn off system-call control using

trustifier set [**extended**] **control off**

This control must be on in order for Trustifier to enforce core security rules.

The only time this control is set to off is for audit-only operations mode.

Use the **extended** command to set the enforcement of Extended Privilege flags on users. See Trustifier Parameters Reference for details on what is enforced

Enforce [extended] mandatory access control on system service requests.

[extended] control

Enforce [extended] mandatory access control on system service requests.

privilege grants

Respect privilege grants and user-trust flags on service requests to non-root users.

success|failure|privilege audits

Audits success, failure and privilege grants on system service requests.

c2 audits

Produce Orange-Book C2 format audit trails.

secrecy

Enforce write-up read-down multi-level security on ordered groups.

integrity

Enforce write-down read-any multi-level data non-repudiation and integrity on ordered groups.

strict groups

User effective group id must be explicitly set to the ordered group for MLS privilege escalation for integrity or secrecy enforcement.

real-uid maps

Use real user-id to lookup access and privilege control. Trustifier uses the effective user-id otherwise.

de-escalation

Set or clear de-escalation locking.

TRUSTIFIER:DB – TRUSTIFIER DATABASE MODULE

Database module for Trustifier Advanced Security System. The commands in the database module control the database.

SYNOPSIS

trustifier load|**restore** [**-d** *FILE-SPEC*] [**-e** *ENCRYPT-ALGO*]

Loads and or restores your persistent permissions into the kernel.

trustifier save [**-d** *FILE-SPEC*] [**-e** *ENCRYPT-ALGO*]

Saves the permissions to disk from the kernel.

trustifier ssn

Displays modules Secure System Number.

Where,

-d Specifies the location for the persistent database.

-e Specifies the encryption algorithm to use. By default the DB module uses AES-256

DESCRIPTION

The Trustifier Database module is used to save and restore Trustifier state to and from the persistent storage.

The database is loaded automatically when /sbin/trustifier is run as the init process. Please see *Trustifier Installation Guide* for more details.

Trustifier DB persistent database default location is:

/etc/trustifier/trustifier.db

TRUSTIFIER:USER – TRUSTIFIER USER MODULE

User module for Trustifier Advanced Security System. The user module commands specify security settings for users, groups and system.

QUICK REFERENCE

Here are some quick examples:

trustifier show jill

Show current settings for user 'jill'

trustifier deny setuid to bob

Deny setuid system call to user bob

trustifier revoke sys_time for root

Revoke the rights to change system time for user root

trustifier audit failed $(trustifier list calls) for @all default

Audit all failed system calls for everyone on the system including unknown users.

trustifier rank bill %engineers=5s

Assign a relative rank of 5s to the user 'bill'

SYNOPSIS

trustifier show [**--xml**] *SUBJECT-SPEC*

Show security settings entity specified by *SUBJECT-SPEC*. Optionally printing output in XML.

trustifier register|**unregister** *SUBJECT-SPEC*

Ensure that a user specified by USER-SPEC or GROUP-SPEC has unique security settings (rather than using group or system defaults).

trustifier allow|**deny** *SERVICE-SPEC* **to** *SUBJECT-SPEC*

Allow or deny services specified by *SERVICE-SPEC* calls to entity identified by *SUBJECT-SPEC*.

trustifier grant|**revoke** *PRIVILEGE-SPEC* **to**|**for** *SUBJECT-SPEC*

Grant or revoke privileges specified by *PRIVILEGE-SPEC* for entity identified by *SUBJECT-SPEC*.

trustifier audit|**unaudit**|**noaudit** [**successful**|**failed**] *SERVICE-SPEC* **for** *SUBJECT-SPEC*

Set or clear auditing flag of a particular SYSCALL for ENTITY

trustifier rank *SUBJECT-SPEC* [**as**] *RANK-SPEC*

Assign rank(s) specified by *RANK-SPEC* to entity identified by *SUBJECT-SPEC*.

PARAMETER AND VALUE SPECIFIERS

Trustifier commands, including commands introduced by new modules take several parameter specifications forms. This section lists the common parameter and value specifications, any which are specific to a module are found in the *Parameter and Value Specifications* section of the module.

Replace the parameter concept specifiers with their literal counterparts described here.

SUBJECT-SPEC: Users, Groups, Everyone or System

Any subject of the system, typically users, users, everyone or systems. A subject-spec is either a USER-SPEC or GROUP-SPEC.

NOTE: In the future SUBJECT-SPEC will also include FILE-SPEC, DEV-SPEC, and PROCESS-SPEC.

USER-SPEC

USER-SPEC means a specific user, all users in a group, everyone that has a login account on the system or any arbitrary user not on defined on the system. USER-SPEC has the following grammar:

```
USER-SPEC   :=   USER-SPEC \s+ USER-SPEC
             |   USER-SPEC
             ;
```

The above grammar simply implies that user-specs can be chained separated by a white space.

```
USER-SPEC   := LOGIN-NAME
             |   USER-ID
             |   @GROUP-NAME
             |   @GROUP-ID
             |   @all | @everyone
             |   default
             ;
LOGIN-NAME     :=     string
USER-ID        := number
GROUP-NAME     := string
GROUP-ID       :=   number
```

The above grammar simply implies that USER-SPEC can be replaced with implied tokens on the right.

Where,

LOGIN-NAME is login name of a user. This will apply the command to the user specified by login-name.

USER-ID is the user-id number of a user. This will apply the command to the specific user-id.

Please note that, depending on the context, USER-ID may not necessarily require an account on the system.

GROUP-NAME is group name of a group. This will apply the command to all the users belonging to the group identified by GROUP-NAME.

GROUP-ID is the group-id of group. This will apply the command to all the users belonging to the group identified by GROUP-ID.

The literals **@all** and **@everyone** imply all users on the system.

The literal **default** implies the default settings for the system. The default settings apply to every user who does not have specifically defined settings.

Examples

bob – indicates the user with the login-id 'bob'

@engineers – indicates all users in the 'engineers' group

jill jane @executives – indicates the users with login-ids 'jill' and 'jane' as well as all the users in the 'executives' group.

@all – indicates all of the users on the system.

GROUP-SPEC

GROUP-SPEC means a group or set of groups on the system. GROUP-SPEC applies associates the commands and security specification with the group-id itself. GROUP-SPEC has the following grammar:

```
GROUP-SPEC :=   GROUP-SPEC \s+ GROUP-SPEC
            |   GROUP-SPEC
            ;
GROUP-SPEC :=   %GROUP-NAME
            |   %GROUP-ID
            |   %all
            |   %default
            ;
GROUP-NAME  :=  string
GROUP-ID    :=  number
```

Where,

GROUP-NAME is the group-name of a group. This will apply the command to the group identified by GROUP-NAME.

GROUP-ID is the group id number identifying a group. This will apply the command to the group identified by GROUP-ID.

The literal **%all** implies all of the groups defined on the system.

The literal **%default** implies the group-level default settings of the system.

IMPORTANT: Please note that the semantics of applying commands to a GROUP are not the same as applying the command to all users within a group.

Examples

%engineers – indicates the 'engineers' group itself (not the individual users within the engineers group).

%engineers %executives – indicates the 'engineers' and the 'executives' groups.

SERVICE-SPEC

A service is a system call or a system service. The grammar for service spec is:

```
SERVICE-SPEC    :=      SERVICE-SPEC \s+ SERVICE-SPEC
                | SERVICE-SPEC
                ;
SERVICE-SPEC := number
                |       string
                ;
```

RANK-SPEC

A rank is a relative indications of trust between two entities. See more under *Understanding Ranks*.

A rank is specified as a number or a label, which is translated into a number using the system's rankings table.

```
RANK-SPEC   :=    ENTITY-ID=RANK-VALUE
                ;
ENTITY-ID   :=    USER-SPEC
                | GROUP-SPEC
                ;
RANK-VALUE      :=      number <RANK-CLASS>
                |       string
                ;
RANK-CLASS:= s[ecret]
                |       i[ntegrity]
                ;
```

Examples

bob=3s – indicates a relative rank of 3 in secrecy for the user 'bob'.

%engineers=3s – indicates a relative rank of 3 in secrecy for the group 'engineers'.

@engineers=3s – indicates a relative rank of 3 in secrecy for all users in the group 'engineers'.

jill=top-secret – indicates a relative rank of top-secret for the user 'jill'.

DESCRIPTION

The Trustifier User module provides the core security specification commands. All commands in the user-module set or clear some security feature for a user, group, or system at large.

Viewing current security settings for an entity

To see the security settings for a user, group or system use the **trustifier show** command.

For example, the following is a (truncated) output for the user bob.

```
# trustifier show bob
# Access control and auditing for user `bob'
restart_syscall                0 [n,n,n]  exit                          1 [y,n,n]
fork                           2 [y,n,n]  read                          3 [y,n,n]
write                          4 [y,n,n]  open                          5 [y,n,n]
close                          6 [y,n,n]  waitpid                       7 [y,n,n]
read                           8 [y,n,n]  link                          9 [y,n,n]
                                       :          :
vmsplice                     316 [n,n,n]  move_pages                   317 [n,n,n]
getcpu                       318 [n,n,n]  epoll_pwait                  319 [n,n,n]

# Capabilities map for user `bob'
chown           [n]    dac_override          [n]    dac_read_search         [n]
fowner          [n]    fsetid                [n]    kill                    [n]
                                       :          :
mknod           [n]    lease                 [n]
# Trust Flags for user `bob'
SO_SOCKET       [y]    SO_CONNECT            [y]    SO_LISTEN               [y]
SO_BIND         [y]    ST_UNKNOWN            [y]    ST_STREAM               [y]
ST_DATAGRAM     [y]    ST_RAW                [y]    ST_RDM                  [y]
                                       :          :
# Ranks for user `bob'
engineers ( 200):  3s;

# Trust factor for user `bob'
trust-factors [cur/max]: 150 / 150
```

The output is divided into four sections:

1. Access Control and Auditing
2. Privileges Control
3. Ranks
4. Internal trust factors

Viewing Access Control and Auditing Settings

The access control and auditing view outputs three fields showing whether a service is allowed, audited for success and audited for failure or not.

For instance, in the diagram above the open system call is being allowed to the user (Allow field is set to 'y') and neither success nor failure of the system call is being recorded for auditing purposes.

Viewing Privileges

Similar to access control output, privilege controls indicate whether or not a privilege is granted to a user:

Viewing Ranks

Rank for a user in relation to an entity is listed as shown below:

Internal Trust Factors

Internal trust factors are internal values that a system assigns to a user. These are not adjustable.

For more information on internal trust factors, please see *Understanding Decider* and the *Decider Reference*.

Putting controls on an entity

Primary method of denying someone access to a particular subsystem is to deny them the system calls that access that subsystem.

For example to deny bob from using setuid subsystem would generally look like:

trustifier deny setuid setuid32 setreuid setreuid32 setresuid setresuid32 to bob

This can be cumbersome to specify. You can use simple shell tools to help, for instance:

trustifier deny $(trustifier list calls| egrep set(re)?s?uid(32)?) to bob

Note that there is a friendlier method that helps with complexity and uses predefined templates that apply to common subsystems. See *Using Templates* later.

Limits extensions

*Please note that the following commands are only available if you have the **limits** module installed.*

The limit module provides extensions to allow and deny commands. It provides the predicates **allow-when, deny-when,** and other limits to allow and deny commands.

You can limit the allowance or denial of a system call based on parametric conditions.

trustifier allow setuid setuid32 setreuid setreuid32 setresuid setresuid32 when uid=1 ruid=1 euid=1 for bob

trustifier deny socketcall to bob when interface=eth0

trustifier limit allow socketcall for bob to 1 request

Auditing an entity

When the system auditing flags are on, then using entity auditing flags you can watch any system call.

Auditing has two distinct generation events, successful call, or failed call. Failure can occur for many reasons, including permission errors.

You can audit a particular system call for a given user using the audit syntax. For example,

trustifier audit failed chroot for ftpuser

trustifier audit successful create_module for root

Granting capabilities (privileges) to an entity

Privileges are extra capabilities traditionally only available to system administrator or root account. Trustifier allows you to assign privileges to non-root users.

For a complete set of grantable capabilities, see the *Trustifier Capabilities and Privileges Reference*.

Grant privileges to an entity as follows:

trustifier grant sys_time to timekeeper

trustifier grant sys_net_bind to httpserv

Limits extensions

*Please note that the following commands are only available if you have the **limits** module installed.*

You can assign a limit on how many times a particular capability may be used by a user.

trustifier grant sys_net_bind to httpserv for 1 request

You may also limit use of a particular privilege to specific system call conditions

trustifier grant sys_net_bind to httpserv for 1 request when syscall=socketcall call=bind port=80

Assigning Trust Ranks to an entity

Trustifier uses Trust Ranks to provide several security services.

Group-wise Inter-User Trust Ranks

Group-wise inter-user trust ranks and Group-wise inter-group trust ranks are used to implement Multi-Level Security (MLS) and Domain Separation (DS).

Group-wise inter-user trust is simply the rank of a user in a group. If a subject (user) of outranks the owner of an object in the group ownership of the object then MAC permissions are granted to the subject.

Group-wise Intern-User Secrecy Ranking (MLS)

Consider a file "secret-recipe" owned by user jill and the group "cola", where user 'jill' has a secrecy ranking. Then, if another user bob has an equal or higher secrecy rank in the group cola than jill, then bob can access "secret-recipe" otherwise not. This is write-up read-down (WURD) MLS, similar to Bell-La Padula labeled security.

To implement simple WURD MLS within a group (say 'secrets') simply give each user (say bob, jill jack and jane) of that group a secrecy ranking:

trustifier rank bob %secrets=3s

trustifier rank jill %secrets=2s

trustifier rank jack %secrets=2s

trustifier rank jane %secrets=1s

The above gives bob the highest ranking within the secrets group and jane the lowest.

Group-wise Inter-User Integrity Ranking (Canon)

Integrity ranking allows the system to maintain and enforce ownership and protect information modification from lower level users. This is equivalent of Biba Labeled security.

Typically integrity ranking is used to ensure non-repudiation and immutability.

To implement simple Biba Labeled security within a group (say 'authoritative') simply give each user (say bob and jane) of that group an integrity ranking:

trustifier rank bob %secrets=3i

trustifier rank jill %secrets=2i

Group-wise Inter-Group rankings

Group-wise Inter-group rankings determine information flow control between two groups.

Trivially any user of the subject group can receive information if the subject group outranks the owner of the information within the object group. There are limitations however, read about it in *Understanding Ranks.*

To assign a group a rank in another group use something similar to:

trustifier rank %subjgrp %objgrp1=3s %objgrp2=2i

Here you are assigning subjgrp a rank of 3s in objgrp1 and 2i in objgrp2.
PLEASE note that Inter-group rankings are asymmetric.

User-wise Inter-User Trust

User-wise inter-user trust is used to create user to user information sharing vectors.

The Read *Understanding Ranks* to learn more.

To allow two users to share information of up to a certain clearance regardless of group ownership of the information source assign an inter-trust between them.

So let's say you want jill to be able to access any files owned by bob within a group where bob's rank is say 3s or below, then assign jill an inter-user trust of 3s for bob:

```
trustifier rank jill bob=3s
```

Internal Trust Factors

Internal Trust Factors are used by the Decider QRAC engine to determine in real-time the Trust the system places on the user.

For further information see the *Decider Reference.*

TRUSTIFIER:TEMPLATES – TRUSTIFIER TEMPLATES MODULE

Trustifier **templates** module provides the **apply** command that can help you build and use predefined security settings that you can apply to any entity.

SYNOPSIS

trustifier list templates

Lists currently defined templates

trustifier apply [**opposite of**] TEMPLATE-SPEC|ENTITY-SPEC [**union**|**intersection** [**opposite of**] TEMPLATE-SPEC|ENTITY-SPEC [...]] **to** ENTITY-SPEC|TEMPLATE-SPEC

Applies settings of one or more templates or entities to another template or entity.

Where,

TEMPLATE-SPEC is a template name or path to a file.

union Implies logical OR of the source with the target.

intersection Implies logical AND of the source with the target.

opposite of Implies logical NOT of the source before performing the apply operation.

Please note that apply supports replacing **union** with **or**, **intersection** with **and**, and **opposite of** with **not**.

DESCRIPTION

You can use the template module to build and store templates that can be applied to each other to build more templates.