

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. |

| Convention | Description |
|------------|--|
| [] | 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 [<i><options></i>] indicates that optional parameters could be typed. |
| | 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 | Refer to the front-end manager program for Trustifier |
| 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, let's 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. It 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 \leftrightarrow object relative trust ranking exists.

| | | SUBJECT RANK (R _s) | | | | |
|---|------------------------------|--------------------------------|---------------------------------|-----------------------|---------------------------------|----------------------|
| | | NO RANKING | SECREC Y RANKING | | INTEGRITY RANKING | |
| OBJ ECT RAN K (R _o) | NO RANK | R,W,X,C | R,X | | R,X | |
| | SECR ECY RANKI NG | NO ACCESS | RANK COMPARISON | ALLOWED OPERATIONS | NO ACCESS | |
| | | | R _u < R _o | NO ACCESS | | |
| | | | R _u = R _o | R,W,X,C | | |
| | | | R _u > R _o | R,X | | |
| | INTEG RITY RANKI NG | R,X | R,X | | RANK COMPARISON | ALLOWED OPERATION |
| | | | | | R _u < R _o | R,X |
| | | | | | R _u = R _o | R,W,X,C |
| R _u > R _o | | | | | R,W,X | |

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 $R(\text{bill}, \text{ted}) \geq 3$. Other conditions may apply before flow of information may be allowed.

The relationship is asymmetric, that is to say:

$$R(u_1, u_2) \neq R(u_2, u_1)$$

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 users¹. 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 transition into mandatory access control very manageable and affordable, in time, effort and resources spent.

¹ 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.

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*.

