Linux Netlabel Documentation

The kernel development community

Jun 10, 2024

CONTENTS

CHAPTER

ONE

NETLABEL INTRODUCTION

Paul Moore, paul.moore@hp.com

August 2, 2006

1.1 Overview

NetLabel is a mechanism which can be used by kernel security modules to attach security attributes to outgoing network packets generated from user space applications and read security attributes from incoming network packets. It is composed of three main components, the protocol engines, the communication layer, and the kernel security module API.

1.2 Protocol Engines

The protocol engines are responsible for both applying and retrieving the network packet's security attributes. If any translation between the network security attributes and those on the host are required then the protocol engine will handle those tasks as well. Other kernel subsystems should refrain from calling the protocol engines directly, instead they should use the NetLabel kernel security module API described below.

Detailed information about each NetLabel protocol engine can be found in this directory.

1.3 Communication Layer

The communication layer exists to allow NetLabel configuration and monitoring from user space. The NetLabel communication layer uses a message based protocol built on top of the Generic NETLINK transport mechanism. The exact formatting of these NetLabel messages as well as the Generic NETLINK family names can be found in the 'net/netlabel/' directory as comments in the header files as well as in 'include/net/netlabel.h'.

1.4 Security Module API

The purpose of the NetLabel security module API is to provide a protocol independent interface to the underlying NetLabel protocol engines. In addition to protocol independence, the security module API is designed to be completely LSM independent which should allow multiple LSMs to leverage the same code base.

Detailed information about the NetLabel security module API can be found in the 'include/net/netlabel.h' header file as well as the '*NetLabel Linux Security Module Interface*' file found in this directory.

CHAPTER

TWO

NETLABEL CIPSO/IPV4 PROTOCOL ENGINE

Paul Moore, paul.moore@hp.com

May 17, 2006

2.1 Overview

The NetLabel CIPSO/IPv4 protocol engine is based on the IETF Commercial IP Security Option (CIPSO) draft from July 16, 1992. A copy of this draft can be found in this directory (draft-ietf-cipso-ipsecurity-01.txt). While the IETF draft never made it to an RFC standard it has become a de-facto standard for labeled networking and is used in many trusted operating systems.

2.2 Outbound Packet Processing

The CIPSO/IPv4 protocol engine applies the CIPSO IP option to packets by adding the CIPSO label to the socket. This causes all packets leaving the system through the socket to have the CIPSO IP option applied. The socket's CIPSO label can be changed at any point in time, however, it is recommended that it is set upon the socket's creation. The LSM can set the socket's CIPSO label by using the NetLabel security module API; if the NetLabel "domain" is configured to use CIPSO for packet labeling then a CIPSO IP option will be generated and attached to the socket.

2.3 Inbound Packet Processing

The CIPSO/IPv4 protocol engine validates every CIPSO IP option it finds at the IP layer without any special handling required by the LSM. However, in order to decode and translate the CIPSO label on the packet the LSM must use the NetLabel security module API to extract the security attributes of the packet. This is typically done at the socket layer using the 'socket_sock_rcv_skb()' LSM hook.

2.4 Label Translation

The CIPSO/IPv4 protocol engine contains a mechanism to translate CIPSO security attributes such as sensitivity level and category to values which are appropriate for the host. These mappings are defined as part of a CIPSO Domain Of Interpretation (DOI) definition and are configured through the NetLabel user space communication layer. Each DOI definition can have a different security attribute mapping table.

2.5 Label Translation Cache

The NetLabel system provides a framework for caching security attribute mappings from the network labels to the corresponding LSM identifiers. The CIPSO/IPv4 protocol engine supports this caching mechanism.

CHAPTER

THREE

NETLABEL LINUX SECURITY MODULE INTERFACE

Paul Moore, paul.moore@hp.com

May 17, 2006

3.1 Overview

NetLabel is a mechanism which can set and retrieve security attributes from network packets. It is intended to be used by LSM developers who want to make use of a common code base for several different packet labeling protocols. The NetLabel security module API is defined in 'include/net/netlabel.h' but a brief overview is given below.

3.2 NetLabel Security Attributes

Since NetLabel supports multiple different packet labeling protocols and LSMs it uses the concept of security attributes to refer to the packet's security labels. The NetLabel security attributes are defined by the 'netlbl_lsm_secattr' structure in the NetLabel header file. Internally the NetLabel subsystem converts the security attributes to and from the correct low-level packet label depending on the NetLabel build time and run time configuration. It is up to the LSM developer to translate the NetLabel security attributes into whatever security identifiers are in use for their particular LSM.

3.3 NetLabel LSM Protocol Operations

These are the functions which allow the LSM developer to manipulate the labels on outgoing packets as well as read the labels on incoming packets. Functions exist to operate both on sockets as well as the sk_buffs directly. These high level functions are translated into low level protocol operations based on how the administrator has configured the NetLabel subsystem.

3.4 NetLabel Label Mapping Cache Operations

Depending on the exact configuration, translation between the network packet label and the internal LSM security identifier can be time consuming. The NetLabel label mapping cache is a caching mechanism which can be used to sidestep much of this overhead once a mapping has been established. Once the LSM has received a packet, used NetLabel to decode its security attributes, and translated the security attributes into a LSM internal identifier the LSM can use the NetLabel caching functions to associate the LSM internal identifier with the network packet's label. This means that in the future when a incoming packet matches a cached value not only are the internal NetLabel translation mechanisms bypassed but the LSM translation mechanisms are bypassed as well which should result in a significant reduction in overhead.

CHAPTER

FOUR

DRAFT IETF CIPSO IP SECURITY

```
IETF CIPSO Working Group
16 July, 1992



                 COMMERCIAL IP SECURITY OPTION (CIPSO 2.2)



1.     Status

This Internet Draft provides the high level specification for a␣
 ↪Commercial
IP Security Option (CIPSO).  This draft reflects the version as␣
 ↪approved by
the CIPSO IETF Working Group.  Distribution of this memo is unlimited.

This document is an Internet Draft.  Internet Drafts are working␣
 ↪documents
of the Internet Engineering Task Force (IETF), its Areas, and its␣
 ↪Working
Groups. Note that other groups may also distribute working documents as
Internet Drafts.

Internet Drafts are draft documents valid for a maximum of six months.
Internet Drafts may be updated, replaced, or obsoleted by other␣
 ↪documents
at any time.  It is not appropriate to use Internet Drafts as reference
material or to cite them other than as a "working draft" or "work in
progress."

Please check the I-D abstract listing contained in each Internet Draft
directory to learn the current status of this or any other Internet␣
 ↪Draft.




2.     Background
```

```
Currently the Internet Protocol includes two security options.  One of
these options is the DoD Basic Security Option (BSO) (Type 130) which␣
↪allows
IP datagrams to be labeled with security classifications.  This option
provides sixteen security classifications and a variable number of␣
↪handling
restrictions.  To handle additional security information, such as␣
↪security
categories or compartments, another security option (Type 133) exists␣
↪and
is referred to as the DoD Extended Security Option (ESO).  The values␣
↪for
the fixed fields within these two options are administered by the␣
↪Defense
Information Systems Agency (DISA).

Computer vendors are now building commercial operating systems with
mandatory access controls and multi-level security.  These systems are
no longer built specifically for a particular group in the defense or
intelligence communities.  They are generally available commercial␣
↪systems
for use in a variety of government and civil sector environments.

The small number of ESO format codes can not support all the possible
applications of a commercial security option.  The BSO and ESO were
designed to only support the United States DoD.  CIPSO has been␣
↪designed
to support multiple security policies.  This Internet Draft provides␣
↪the
format and procedures required to support a Mandatory Access Control
security policy.  Support for additional security policies shall be
defined in future RFCs.




Internet Draft, Expires 15 Jan 93                                  ␣
↪[PAGE 1]



CIPSO INTERNET DRAFT                                            16 July,␣
↪1992




3.    CIPSO Format
```

```
Option type: 134 (Class 0, Number 6, Copy on Fragmentation)
Option length: Variable

This option permits security related information to be passed between
systems within a single Domain of Interpretation (DOI).  A DOI is a
collection of systems which agree on the meaning of particular values
in the security option.  An authority that has been assigned a DOI
identifier will define a mapping between appropriate CIPSO field values
and their human readable equivalent.  This authority will distribute␣
↪that
mapping to hosts within the authority's domain.  These mappings may be
sensitive, therefore a DOI authority is not required to make these
mappings available to anyone other than the systems that are included␣
↪in
the DOI.

This option MUST be copied on fragmentation.  This option appears at␣
↪most
once in a datagram.  All multi-octet fields in the option are defined␣
↪to be
transmitted in network byte order.  The format of this option is as␣
↪follows:

+----------+----------+------//------+-----------//---------+
| 10000110 | LLLLLLLL | DDDDDDDDDDDD | TTTTTTTTTTTTTTTTTTTT |
+----------+----------+------//------+-----------//---------+

  TYPE=134    OPTION    DOMAIN OF               TAGS
              LENGTH    INTERPRETATION


                Figure 1. CIPSO Format


3.1    Type

This field is 1 octet in length.  Its value is 134.


3.2    Length

This field is 1 octet in length.  It is the total length of the option
including the type and length fields.  With the current IP header␣
↪length
restriction of 40 octets the value of this field MUST not exceed 40.


3.3    Domain of Interpretation Identifier

This field is an unsigned 32 bit integer.  The value 0 is reserved and␣
```

```
↪MUST
not appear as the DOI identifier in any CIPSO option.  Implementations
should assume that the DOI identifier field is not aligned on any␣
↪particular
byte boundary.

To conserve space in the protocol, security levels and categories are
represented by numbers rather than their ASCII equivalent.  This␣
↪requires
a mapping table within CIPSO hosts to map these numbers to their
corresponding ASCII representations.  Non-related groups of systems may



Internet Draft, Expires 15 Jan 93                                      ␣
↪[PAGE 2]




CIPSO INTERNET DRAFT                                             16 July,␣
↪1992




have their own unique mappings.  For example, one group of systems may
use the number 5 to represent Unclassified while another group may use␣
↪the
number 1 to represent that same security level.  The DOI identifier is␣
↪used
to identify which mapping was used for the values within the option.


3.4    Tag Types

A common format for passing security related information is necessary
for interoperability.  CIPSO uses sets of "tags" to contain the␣
↪security
information relevant to the data in the IP packet.  Each tag begins␣
↪with
a tag type identifier followed by the length of the tag and ends with␣
↪the
actual security information to be passed.  All multi-octet fields in a␣
↪tag
are defined to be transmitted in network byte order.  Like the DOI
identifier field in the CIPSO header, implementations should assume␣
↪that
all tags, as well as fields within a tag, are not aligned on any␣
↪particular
octet boundary.   The tag types defined in this document contain␣
↪alignment
```

```
bytes to assist alignment of some information, however alignment can␣
↪not
be guaranteed if CIPSO is not the first IP option.

CIPSO tag types 0 through 127 are reserved for defining standard tag
formats.  Their definitions will be published in RFCs.  Tag types whose
identifiers are greater than 127 are defined by the DOI authority and␣
↪may
only be meaningful in certain Domains of Interpretation.  For these tag
types, implementations will require the DOI identifier as well as the␣
↪tag
number to determine the security policy and the format associated with␣
↪the
tag.  Use of tag types above 127 are restricted to closed networks␣
↪where
interoperability with other networks will not be an issue. ␣
↪Implementations
that support a tag type greater than 127 MUST support at least one DOI␣
↪that
requires only tag types 1 to 127.

Tag type 0 is reserved. Tag types 1, 2, and 5 are defined in this
Internet Draft.  Types 3 and 4 are reserved for work in progress.
The standard format for all current and future CIPSO tags is shown␣
↪below:

+----------+----------+--------//--------+
| TTTTTTTT | LLLLLLLL | IIIIIIIIIIIIIIII |
+----------+----------+--------//--------+
    TAG        TAG         TAG
    TYPE       LENGTH      INFORMATION

    Figure 2:  Standard Tag Format

In the three tag types described in this document, the length and count
restrictions are based on the current IP limitation of 40 octets for␣
↪all
IP options.  If the IP header is later expanded, then the length and␣
↪count
restrictions specified in this document may increase to use the full␣
↪area
provided for IP options.


3.4.1    Tag Type Classes

Tag classes consist of tag types that have common processing␣
↪requirements
and support the same security policy.  The three tags defined in this
Internet Draft belong to the Mandatory Access Control (MAC) Sensitivity
```

```
Internet Draft, Expires 15 Jan 93                                            
↪[PAGE 3]



CIPSO INTERNET DRAFT                                             16 July,
↪1992



class and support the MAC Sensitivity security policy.


3.4.2    Tag Type 1

This is referred to as the "bit-mapped" tag type.  Tag type 1 is
↪included
in the MAC Sensitivity tag type class.  The format of this tag type is
↪as
follows:

+----------+----------+----------+----------+--------//---------+
| 00000001 | LLLLLLLL | 00000000 | LLLLLLLL | CCCCCCCCCCCCCCCCC |
+----------+----------+----------+----------+--------//---------+

    TAG       TAG      ALIGNMENT  SENSITIVITY    BIT MAP OF
    TYPE      LENGTH   OCTET      LEVEL          CATEGORIES

            Figure 3. Tag Type 1 Format


3.4.2.1    Tag Type

This field is 1 octet in length and has a value of 1.


3.4.2.2    Tag Length

This field is 1 octet in length.  It is the total length of the tag
↪type
including the type and length fields.  With the current IP header
↪length
restriction of 40 bytes the value within this field is between 4 and
↪34.


3.4.2.3    Alignment Octet
```

```
This field is 1 octet in length and always has the value of 0.  Its␣
↪purpose
is to align the category bitmap field on an even octet boundary.  This␣
↪will
speed many implementations including router implementations.


3.4.2.4    Sensitivity Level

This field is 1 octet in length.  Its value is from 0 to 255.  The␣
↪values
are ordered with 0 being the minimum value and 255 representing the␣
↪maximum
value.


3.4.2.5    Bit Map of Categories

The length of this field is variable and ranges from 0 to 30 octets. ␣
↪This
provides representation of categories 0 to 239.  The ordering of the␣
↪bits
is left to right or MSB to LSB.  For example category 0 is represented␣
↪by
the most significant bit of the first byte and category 15 is␣
↪represented
by the least significant bit of the second byte.  Figure 4 graphically
shows this ordering.  Bit N is binary 1 if category N is part of the␣
↪label
for the datagram, and bit N is binary 0 if category N is not part of␣
↪the
label.  Except for the optimized tag 1 format described in the next␣
↪section,



Internet Draft, Expires 15 Jan 93                                      ␣
↪[PAGE 4]



CIPSO INTERNET DRAFT                                         16 July,␣
↪1992



minimal encoding SHOULD be used resulting in no trailing zero octets␣
↪in the
category bitmap.
```

```
        octet 0  octet 1  octet 2  octet 3  octet 4  octet 5
        XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX . . .
bit     01234567 89111111 11112222 22222233 33333333 44444444
number             012345 67890123 45678901 23456789 01234567

            Figure 4. Ordering of Bits in Tag 1 Bit Map


3.4.2.6    Optimized Tag 1 Format

Routers work most efficiently when processing fixed length fields.  To
support these routers there is an optimized form of tag type 1.  The␣
↪format
does not change.  The only change is to the category bitmap which is␣
↪set to
a constant length of 10 octets.  Trailing octets required to fill out␣
↪the 10
octets are zero filled.  Ten octets, allowing for 80 categories, was␣
↪chosen
because it makes the total length of the CIPSO option 20 octets.  If␣
↪CIPSO
is the only option then the option will be full word aligned and␣
↪additional
filler octets will not be required.


3.4.3    Tag Type 2

This is referred to as the "enumerated" tag type.  It is used to␣
↪describe
large but sparsely populated sets of categories.  Tag type 2 is in the␣
↪MAC
Sensitivity tag type class.  The format of this tag type is as follows:

+----------+---------+---------+---------+------------//-----------
↪--+
| 00000010 | LLLLLLLL | 00000000 | LLLLLLLL |␣
↪CCCCCCCCCCCCCCCCCCCCCCCCCC |
+----------+---------+---------+---------+------------//-----------
↪--+

    TAG       TAG      ALIGNMENT  SENSITIVITY         ENUMERATED
    TYPE      LENGTH   OCTET      LEVEL               CATEGORIES

                Figure 5. Tag Type 2 Format


3.4.3.1     Tag Type
```

```
This field is one octet in length and has a value of 2.


3.4.3.2    Tag Length

This field is 1 octet in length. It is the total length of the tag type
including the type and length fields.  With the current IP header␣
↪length
restriction of 40 bytes the value within this field is between 4 and␣
↪34.


3.4.3.3    Alignment Octet

This field is 1 octet in length and always has the value of 0.  Its␣
↪purpose
is to align the category field on an even octet boundary.  This will



Internet Draft, Expires 15 Jan 93                                   ␣
↪[PAGE 5]




CIPSO INTERNET DRAFT                                        16 July,␣
↪1992




speed many implementations including router implementations.


3.4.3.4    Sensitivity Level

This field is 1 octet in length. Its value is from 0 to 255.  The␣
↪values
are ordered with 0 being the minimum value and 255 representing the
maximum value.


3.4.3.5    Enumerated Categories

In this tag, categories are represented by their actual value rather␣
↪than
by their position within a bit field.  The length of each category is 2
octets.  Up to 15 categories may be represented by this tag.  Valid␣
↪values
for categories are 0 to 65534.  Category 65535 is not a valid category
value.  The categories MUST be listed in ascending order within the␣
```

```
↪tag.


3.4.4    Tag Type 5

This is referred to as the "range" tag type.  It is used to represent
labels where all categories in a range, or set of ranges, are included
in the sensitivity label.  Tag type 5 is in the MAC Sensitivity tag␣
↪type
class.  The format of this tag type is as follows:

+----------+---------+---------+---------+------------//------------
↪-+
| 00000101 | LLLLLLLL | 00000000 | LLLLLLLL |  Top/Bottom | Top/Bottom␣
↪ |
+----------+---------+---------+---------+------------//------------
↪-+

    TAG       TAG      ALIGNMENT  SENSITIVITY        CATEGORY RANGES
    TYPE      LENGTH   OCTET      LEVEL

                     Figure 6. Tag Type 5 Format


3.4.4.1     Tag Type

This field is one octet in length and has a value of 5.


3.4.4.2    Tag Length

This field is 1 octet in length. It is the total length of the tag type
including the type and length fields.  With the current IP header␣
↪length
restriction of 40 bytes the value within this field is between 4 and␣
↪34.


3.4.4.3    Alignment Octet

This field is 1 octet in length and always has the value of 0.  Its␣
↪purpose
is to align the category range field on an even octet boundary.  This␣
↪will
speed many implementations including router implementations.
```

```
Internet Draft, Expires 15 Jan 93                                          ␣
↪[PAGE 6]



CIPSO INTERNET DRAFT                                              16 July,␣
↪1992



3.4.4.4    Sensitivity Level

This field is 1 octet in length. Its value is from 0 to 255.  The␣
↪values
are ordered with 0 being the minimum value and 255 representing the␣
↪maximum
value.



3.4.4.5    Category Ranges

A category range is a 4 octet field comprised of the 2 octet index of␣
↪the
highest numbered category followed by the 2 octet index of the lowest
numbered category.  These range endpoints are inclusive within the␣
↪range of
categories.  All categories within a range are included in the␣
↪sensitivity
label.  This tag may contain a maximum of 7 category pairs.  The bottom
category endpoint for the last pair in the tag MAY be omitted and␣
↪SHOULD be
assumed to be 0.  The ranges MUST be non-overlapping and be listed in
descending order.  Valid values for categories are 0 to 65534. ␣
↪Category
65535 is not a valid category value.



3.4.5     Minimum Requirements

A CIPSO implementation MUST be capable of generating at least tag type␣
↪1 in
the non-optimized form.  In addition, a CIPSO implementation MUST be␣
↪able
to receive any valid tag type 1 even those using the optimized tag␣
↪type 1
format.



4.    Configuration Parameters
```

```
The configuration parameters defined below are required for all CIPSO␣
↪hosts,
gateways, and routers that support multiple sensitivity labels.  A␣
↪CIPSO
host is defined to be the origination or destination system for an IP
datagram.  A CIPSO gateway provides IP routing services between two or␣
↪more
IP networks and may be required to perform label translations between
networks.  A CIPSO gateway may be an enhanced CIPSO host or it may just
provide gateway services with no end system CIPSO capabilities.  A␣
↪CIPSO
router is a dedicated IP router that routes IP datagrams between two␣
↪or more
IP networks.

An implementation of CIPSO on a host MUST have the capability to␣
↪reject a
datagram for reasons that the information contained can not be␣
↪adequately
protected by the receiving host or if acceptance may result in␣
↪violation of
the host or network security policy.  In addition, a CIPSO gateway or␣
↪router
MUST be able to reject datagrams going to networks that can not provide
adequate protection or may violate the network's security policy.  To
provide this capability the following minimal set of configuration
parameters are required for CIPSO implementations:

HOST_LABEL_MAX - This parameter contains the maximum sensitivity label␣
↪that
a CIPSO host is authorized to handle.  All datagrams that have a label
greater than this maximum MUST be rejected by the CIPSO host.  This
parameter does not apply to CIPSO gateways or routers.  This parameter␣
↪need
not be defined explicitly as it can be implicitly derived from the
PORT_LABEL_MAX parameters for the associated interfaces.



Internet Draft, Expires 15 Jan 93                                   ␣
↪[PAGE 7]



CIPSO INTERNET DRAFT                                          16 July,␣
↪1992
```

```
HOST_LABEL_MIN - This parameter contains the minimum sensitivity label␣
↪that
a CIPSO host is authorized to handle.  All datagrams that have a label␣
↪less
than this minimum MUST be rejected by the CIPSO host.  This parameter␣
↪does
not apply to CIPSO gateways or routers.  This parameter need not be␣
↪defined
explicitly as it can be implicitly derived from the PORT_LABEL_MIN
parameters for the associated interfaces.

PORT_LABEL_MAX - This parameter contains the maximum sensitivity label␣
↪for
all datagrams that may exit a particular network interface port.  All
outgoing datagrams that have a label greater than this maximum MUST be
rejected by the CIPSO system.  The label within this parameter MUST be
less than or equal to the label within the HOST_LABEL_MAX parameter. ␣
↪This
parameter does not apply to CIPSO hosts that support only one network␣
↪port.

PORT_LABEL_MIN - This parameter contains the minimum sensitivity label␣
↪for
all datagrams that may exit a particular network interface port.  All
outgoing datagrams that have a label less than this minimum MUST be
rejected by the CIPSO system.  The label within this parameter MUST be
greater than or equal to the label within the HOST_LABEL_MIN parameter.
This parameter does not apply to CIPSO hosts that support only one␣
↪network
port.

PORT_DOI - This parameter is used to assign a DOI identifier value to a
particular network interface port.  All CIPSO labels within datagrams
going out this port MUST use the specified DOI identifier.  All CIPSO
hosts and gateways MUST support either this parameter, the NET_DOI
parameter, or the HOST_DOI parameter.

NET_DOI - This parameter is used to assign a DOI identifier value to a
particular IP network address.  All CIPSO labels within datagrams␣
↪destined
for the particular IP network MUST use the specified DOI identifier. ␣
↪All
CIPSO hosts and gateways MUST support either this parameter, the PORT_
↪DOI
parameter, or the HOST_DOI parameter.

HOST_DOI - This parameter is used to assign a DOI identifier value to a
particular IP host address.  All CIPSO labels within datagrams␣
↪destined for
the particular IP host will use the specified DOI identifier.  All␣
```

```
↪CIPSO
hosts and gateways MUST support either this parameter, the PORT_DOI
parameter, or the NET_DOI parameter.

This list represents the minimal set of configuration parameters␣
↪required
to be compliant.  Implementors are encouraged to add to this list to
provide enhanced functionality and control.  For example, many security
policies may require both incoming and outgoing datagrams be checked␣
↪against
the port and host label ranges.


4.1    Port Range Parameters

The labels represented by the PORT_LABEL_MAX and PORT_LABEL_MIN␣
↪parameters
MAY be in CIPSO or local format.  Some CIPSO systems, such as routers,␣
↪may
want to have the range parameters expressed in CIPSO format so that␣
↪incoming
labels do not have to be converted to a local format before being␣
↪compared
against the range.  If multiple DOIs are supported by one of these␣
↪CIPSO



Internet Draft, Expires 15 Jan 93                                     ␣
↪[PAGE 8]



CIPSO INTERNET DRAFT                                         16 July,␣
↪1992



systems then multiple port range parameters would be needed, one set␣
↪for
each DOI supported on a particular port.

The port range will usually represent the total set of labels that may
exist on the logical network accessed through the corresponding network
interface.  It may, however, represent a subset of these labels that␣
↪are
allowed to enter the CIPSO system.


4.2    Single Label CIPSO Hosts
```

```
CIPSO implementations that support only one label are not required to
support the parameters described above.  These limited implementations␣
↪are
only required to support a NET_LABEL parameter.  This parameter␣
↪contains
the CIPSO label that may be inserted in datagrams that exit the host. ␣
↪In
addition, the host MUST reject any incoming datagram that has a label␣
↪which
is not equivalent to the NET_LABEL parameter.


5.    Handling Procedures

This section describes the processing requirements for incoming and
outgoing IP datagrams.  Just providing the correct CIPSO label format
is not enough.  Assumptions will be made by one system on how a
receiving system will handle the CIPSO label.  Wrong assumptions may
lead to non-interoperability or even a security incident.  The
requirements described below represent the minimal set needed for
interoperability and that provide users some level of confidence.
Many other requirements could be added to increase user confidence,
however at the risk of restricting creativity and limiting vendor
participation.


5.1    Input Procedures

All datagrams received through a network port MUST have a security␣
↪label
associated with them, either contained in the datagram or assigned to␣
↪the
receiving port.  Without this label the host, gateway, or router will␣
↪not
have the information it needs to make security decisions.  This␣
↪security
label will be obtained from the CIPSO if the option is present in the
datagram.  See section 4.1.2 for handling procedures for unlabeled
datagrams.  This label will be compared against the PORT (if␣
↪appropriate)
and HOST configuration parameters defined in section 3.

If any field within the CIPSO option, such as the DOI identifier, is␣
↪not
recognized the IP datagram is discarded and an ICMP "parameter problem"
(type 12) is generated and returned.  The ICMP code field is set to
↪"bad
parameter" (code 0) and the pointer is set to the start of the CIPSO␣
↪field
```

```
that is unrecognized.

If the contents of the CIPSO are valid but the security label is
outside of the configured host or port label range, the datagram is
discarded and an ICMP "destination unreachable" (type 3) is generated
and returned.  The code field of the ICMP is set to "communication with
destination network administratively prohibited" (code 9) or to



Internet Draft, Expires 15 Jan 93                                         ␣
↪[PAGE 9]



CIPSO INTERNET DRAFT                                              16 July,␣
↪1992



"communication with destination host administratively prohibited"
(code 10).  The value of the code field used is dependent upon whether
the originator of the ICMP message is acting as a CIPSO host or a CIPSO
gateway.  The recipient of the ICMP message MUST be able to handle␣
↪either
value.  The same procedure is performed if a CIPSO can not be added to␣
↪an
IP packet because it is too large to fit in the IP options area.

If the error is triggered by receipt of an ICMP message, the message
is discarded and no response is permitted (consistent with general ICMP
processing rules).


5.1.1    Unrecognized tag types

The default condition for any CIPSO implementation is that an
unrecognized tag type MUST be treated as a "parameter problem" and
handled as described in section 4.1.  A CIPSO implementation MAY allow
the system administrator to identify tag types that may safely be
ignored.  This capability is an allowable enhancement, not a
requirement.


5.1.2    Unlabeled Packets

A network port may be configured to not require a CIPSO label for all
incoming  datagrams.  For this configuration a CIPSO label must be
assigned to that network port and associated with all unlabeled IP
datagrams.  This capability might be used for single level networks or
```

```
networks that have CIPSO and non-CIPSO hosts and the non-CIPSO hosts
all operate at the same label.

If a CIPSO option is required and none is found, the datagram is
discarded and an ICMP "parameter problem" (type 12) is generated and
returned to the originator of the datagram.  The code field of the ICMP
is set to "option missing" (code 1) and the ICMP pointer is set to 134
(the value of the option type for the missing CIPSO option).


5.2    Output Procedures

A CIPSO option MUST appear only once in a datagram.  Only one tag type
from the MAC Sensitivity class MAY be included in a CIPSO option. 
↪Given
the current set of defined tag types, this means that CIPSO labels at
first will contain only one tag.

All datagrams leaving a CIPSO system MUST meet the following condition:

        PORT_LABEL_MIN <= CIPSO label <= PORT_LABEL_MAX

If this condition is not satisfied the datagram MUST be discarded.
If the CIPSO system only supports one port, the HOST_LABEL_MIN and the
HOST_LABEL_MAX parameters MAY be substituted for the PORT parameters in
the above condition.

The DOI identifier to be used for all outgoing datagrams is configured
↪by



Internet Draft, Expires 15 Jan 93                                     
↪[PAGE 10]




CIPSO INTERNET DRAFT                                         16 July,
↪1992




the administrator.  If port level DOI identifier assignment is used,
↪then
the PORT_DOI configuration parameter MUST contain the DOI identifier to
use.  If network level DOI assignment is used, then the NET_DOI
↪parameter
MUST contain the DOI identifier to use.  And if host level DOI
↪assignment
is employed, then the HOST_DOI parameter MUST contain the DOI
```

```
↪identifier
to use.  A CIPSO implementation need only support one level of DOI
assignment.


5.3    DOI Processing Requirements

A CIPSO implementation MUST support at least one DOI and SHOULD support
multiple DOIs.  System and network administrators are cautioned to
ensure that at least one DOI is common within an IP network to allow␣
↪for
broadcasting of IP datagrams.

CIPSO gateways MUST be capable of translating a CIPSO option from one
DOI to another when forwarding datagrams between networks.  For
efficiency purposes this capability is only a desired feature for CIPSO
routers.


5.4    Label of ICMP Messages

The CIPSO label to be used on all outgoing ICMP messages MUST be␣
↪equivalent
to the label of the datagram that caused the ICMP message.  If the␣
↪ICMP was
generated due to a problem associated with the original CIPSO label␣
↪then the
following responses are allowed:

  a.  Use the CIPSO label of the original IP datagram
  b.  Drop the original datagram with no return message generated

In most cases these options will have the same effect.  If you can not
interpret the label or if it is outside the label range of your host or
interface then an ICMP message with the same label will probably not be
able to exit the system.


6.    Assignment of DOI Identifier Numbers                              ␣
↪      =

Requests for assignment of a DOI identifier number should be addressed␣
↪to
the Internet Assigned Numbers Authority (IANA).


7.    Acknowledgements

Much of the material in this RFC is based on (and copied from) work
done by Gary Winiger of Sun Microsystems and published as Commercial
```

```
IP Security Option at the INTEROP 89, Commercial IPSO Workshop.


8.    Author's Address

To submit mail for distribution to members of the IETF CIPSO Working
Group, send mail to: cipso@wdl1.wdl.loral.com.




Internet Draft, Expires 15 Jan 93
↪[PAGE 11]




CIPSO INTERNET DRAFT                                              16 July,
↪1992





To be added to or deleted from this distribution, send mail to:
cipso-request@wdl1.wdl.loral.com.


9.    References

RFC 1038, "Draft Revised IP Security Option", M. St. Johns, IETF,
↪January
1988.

RFC 1108, "U.S. Department of Defense Security Options
for the Internet Protocol", Stephen Kent, IAB, 1 March, 1991.
```

```
Internet Draft, Expires 15 Jan 93
→[PAGE 12]
```