#### Security Policy Object

The SecurityPolicy object consists of a list of rules, each of which has:

1. UUID
2. IP protocol
3. One src IP address range or prefix
4. One dst IP address range or prefix
5. One source **port range** (or ICMP type/code)
6. One destination **port range** (or ICMP type/code)

If the user facing rule has multiple IP prefixes and/or port ranges, the user facing policy will need to be expanded but the **same uuid as the user facing rule can be re-used across the expanded rules** (as it is used for reporting which user facing rule matched the packet).

**Example:**

User facing rule (rules are shown in json format for ease of understanding, the lines below do not reflect exact protobuf format):

R1: {

“uuid” : “uuid1”,

“ProtoNum” : “6”,

“SrcPrefix”: “1.1.1.0/24, 2.2.2.0/24”,

“DstPrefix”: “3.3.3.0/24, 4.4.4.0/24”,

“SrcPortRange”: “8000-9000, 10000-11000”,

“DstPortRange”: “8080, 20000-25000”,

}

Application agent is expected to expand the rule as follows and send to pdsagent:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UUID** | **Proto** | **SrcPrefix** | **DstPrefix** | **SrcPortRange** | **DstPortRange** |
| uuid1 | 6 | 1.1.1.0/24 | 3.3.3.0/24 | 8000-9000 | 8080 |
| uuid1 | 6 | 1.1.1.0/24 | 3.3.3.0/24 | 8000-9000 | 20000-25000 |
| uuid1 | 6 | 1.1.1.0/24 | 3.3.3.0/24 | 10000-11000 | 8080 |
| uuid1 | 6 | 1.1.1.0/24 | 3.3.3.0/24 | 10000-11000 | 20000-25000 |
| uuid1 | 6 | 1.1.1.0/24 | 4.4.4.0/24” | 8000-9000 | 8080 |
| uuid1 | 6 | 1.1.1.0/24 | 4.4.4.0/24” | 8000-9000 | 20000-25000 |
| uuid1 | 6 | 1.1.1.0/24 | 4.4.4.0/24” | 10000-11000 | 8080 |
| uuid1 | 6 | 1.1.1.0/24 | 4.4.4.0/24” | 10000-11000 | 20000-25000 |
| uuid1 | 6 | 2.2.2.0/24 | 3.3.3.0/24 | 8000-9000 | 8080 |
| uuid1 | 6 | 2.2.2.0/24 | 3.3.3.0/24 | 8000-9000 | 20000-25000 |
| uuid1 | 6 | 2.2.2.0/24 | 3.3.3.0/24 | 10000-11000 | 8080 |
| uuid1 | 6 | 2.2.2.0/24 | 3.3.3.0/24 | 10000-11000 | 20000-25000 |
| uuid1 | 6 | 2.2.2.0/24 | 4.4.4.0/24” | 8000-9000 | 8080 |
| uuid1 | 6 | 2.2.2.0/24 | 4.4.4.0/24” | 8000-9000 | 20000-25000 |
| uuid1 | 6 | 2.2.2.0/24 | 4.4.4.0/24” | 10000-11000 | 8080 |
| uuid1 | 6 | 2.2.2.0/24 | 4.4.4.0/24” | 10000-11000 | 20000-25000 |

**Note that the uuid is repeated in all the expanded rules and all the expanded rules are sharing the same uuid are expected to be contiguous in the list of rules in a given policy.** **All rules sharing the same uuid like this are considered as 1 rule. As of now we are supporting 1k Rules per NSG per direction.**

There are two main resources on the DSC that are optimized:

**Memory**: pdsagent doesn’t maintain an in-memory database for rules or policies (indexed by uuids) after programming the datapath. The config is received, processed and programmed in the datapath and is discarded so there are no two copies of the same given the very high scale of the policies and rules that need to be supported. This will help in supporting higher scale and features like ISSU etc..

**ARM CPU**: The policy expansion is expected to be performed by the application agent external to ARM for multiple reasons. ARM CPU is used to support very high sustained CPS (for flow/session installation with CPS of above 4M per second) and assist in flow aging to delete the expired flows. The other reason is having variable number of prefixes for each match condition in a given rule and across rules will cause memory fragmentation as varied size memory chunks will be allocated and freed and these operations will take a significant number of CPU cycles that can be used for other things on ARM.

#### Compact Policy Protos (Proposal)

// IPEntry represents any form of IP address/prefix/range/tag etc.

**message IPEntry {**

**oneof ip\_entry {**

**// IP prefix**

**IPPrefix Prefix = 1;**

**// IP range**

**AddressRange Range = 2;**

**// tag that represents IP addres/pfx/range**

**uint32 Tag = 3;**

**}**

**}**

**// IPList is a list of IPEntry objects**

**message IPList {**

**repeated IPEntry IPEntry = 1;**

**}**

// L3 rule match criteria

message RuleL3Match {

// IP Protocol

oneof protomatch {

uint32 ProtoNum = 1;

WildcardMatch ProtoWildcard = 2; // Match ANY protocol

}

oneof srcmatch {

IPPrefix SrcPrefix = 3; // Src IP Prefix

AddressRange SrcRange = 4; // Src IP Range

uint32 SrcTag = 5; // Src Tag

**// NOTE: this attribute is not supported**

**IPList SrcIPList = 6; // Src IP List**

}

oneof dstmatch {

IPPrefix DstPrefix = 7; // Dst IP Prefix

AddressRange DstRange = 8; // Dst IP Range

uint32 DstTag = 9; // Dst Tag

**// NOTE: this attribute is not supported**

**IPList DstIPList = 10; // Dst IP List**

}

}

// TCP/UDP rule match criteria

message PortMatch {

// source port range

PortRange SrcPortRange = 1;

// destination port range

PortRange DstPortRange = 2;

}

// ICMPv4/ICMPv6 rule match criteria

message ICMPMatch {

// ICMP type

oneof typematch {

// match any ICMP type

uint32 Type = 1;

// match any ICMP type

WildcardMatch TypeWildcard = 2;

}

// ICMP code

oneof codematch {

// match any ICMP code

uint32 Code = 3;

// match any ICMP code

WildcardMatch CodeWildcard = 4;

}

}

// TCP/UDP source and destination port list

message PortListMatch {

// list of source ports or port ranges

repeated PortRange SrcPortRange = 1;

// list of destination ports or port ranges

repeated PortRange DstPortRange = 2;

}

// ICMP type/code match condition list

message ICMPMatchList {

// ICMP type/code list

repeated ICMPMatch ICMPMatchList = 1;

}

// L4 rule match criteria

message RuleL4Match {

oneof l4info {

// source and/or destination ports/ranges

PortMatch Ports = 1;

// ICMP type/code match criteria

ICMPMatch TypeCode = 2;

**// list of source and/or destination ports/ranges**

**// NOTE: this attribute is not supported**

**PortListMatch PortList = 3;**

**// list ICMP type/code match criteria**

**// NOTE: this attribute is not supported**

**ICMPMatchList ICMPMatchList = 4;**

}

}

// rule match criteria

message RuleMatch {

// Layer 3 match criteria

RuleL3Match L3Match = 1;

// Layer 4 match criteria

RuleL4Match L4Match = 2;

}