Using CDB lists

OSSEC is able to check if a field extracted during the decoding phase is in a CDB list (constant database). The main use case of this feature is to create a white/black list of users, IPs or domain names.

Creating a CDB list

Creating the list file

The list file is a plain text file where each line has the following format:

```
key1:value1
key2:value2
```

Each key must be unique and is terminated with a colon `:`.

For IP addresses the dot notation is used for subnet matches:

key	CIDR	Possible matches
192.168.:	192.168.0.0/16	192.168.0.0 - 192.168.255.255
172.16.19.:	172.16.19.0/24	172.16.19.0 - 172.16.19.255
10.1.1.1:	10.1.1.1/32	10.1.1.1

Example of IP address list file:

```
192.168.: Matches 192.168.0.0 - 192.168.255.255
172.16.19.: Matches 172.16.19.0 - 172.16.19.255
10.1.1.1: Matches 10.1.1.1
```

We recommend to use the directory */var/ossec/etc/lists* to store your lists.

Adding the list to ossec.conf

Each list must be defined in the ossec.conf file using the following syntax:

```
<ossec_config>
    <ruleset>
        <list>etc/lists/list-IP</list>
```

Restart OSSEC to apply the changes.

Making the CDB list

The list files must be compiled before they can be used. The tool */var/ossec/bin/ossec-makelists* will process and compile all the lists if needed.

Remember to compile the lists every time that you update them. It is not necessary to restart OSSEC to apply the changes.

Using the CDB list in the rules

A rule would use the following syntax to look up a key within a CDB list.

Positive key match

This example is a search for the key stored in the field attribute and will match if it *IS* present in the database:

```
<list field="user" lookup="match_key">etc/lists/list-user</list>
```

The `lookup="match_key"` is the default and can be left out as in this example:

```
<list field="user">etc/lists/list-user</list>
```

In case the field is an IP address, you must to use *address_match_key*:

```
<list field="srcip" lookup="address_match_key">etc/lists/list-IP</list>
```

Negative key match

This example is a search for the key stored in the field attribute and will match if it *IS NOT* present in the database:

```
<list field="user" lookup="not_match_key">etc/lists/list-user</list>
```

In case the field is an IP address, you must use *not_address_match_key*:

```
<list field="srcip" lookup="not_address_match_key">etc/lists/list-IP</list>
```

Key and value match

This example is a search for the key stored in the field attribute, and on a positive match the returned value of the key will be processed using the regex in the check_value attribute:

```
<list field="user" lookup="match_key_value" check_value="^block">etc/lists/list-user</list>
```

In case the field is an IP address, you must use *not_address_match_key*:

```
<list field="srcip" lookup="address_match_key_value" check_value="^reject">etc/lists/list-IP</list>
```