How To Configure and Tune CoreXL on SecurePlatform

10 April 2012
© 2012 Check Point Software Technologies Ltd.

All rights reserved. This product and related documentation are protected by copyright and distributed under licensing restricting their use, copying, distribution, and decompilation. No part of this product or related documentation may be reproduced in any form or by any means without prior written authorization of Check Point. While every precaution has been taken in the preparation of this book, Check Point assumes no responsibility for errors or omissions. This publication and features described herein are subject to change without notice.

RESTRICTED RIGHTS LEGEND:

Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

TRADEMARKS:

Refer to the Copyright page (http://www.checkpoint.com/copyright.html) for a list of our trademarks.

Refer to the Third Party copyright notices (http://www.checkpoint.com/3rd_party_copyright.html) for a list of relevant copyrights and third-party licenses.
Important Information

Latest Software
We recommend that you install the most recent software release to stay up-to-date with the latest functional improvements, stability fixes, security enhancements and protection against new and evolving attacks.

Latest Documentation
The latest version of this document is at:
http://supportcontent.checkpoint.com/documentation_download?ID=14884

For additional technical information, visit the Check Point Support Center (http://supportcenter.checkpoint.com).

Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4/10/2012</td>
<td>First release of this document</td>
</tr>
</tbody>
</table>

Feedback
Check Point is engaged in a continuous effort to improve its documentation.
Please help us by sending your comments (mailto:cp_techpub_feedback@checkpoint.com?subject=Feedback on How To Configure and Tune CoreXL on SecurePlatform ).
How To Configure and Tune CoreXL on SecurePlatform

Objective

This document explains what CoreXL is, what it achieves, and how to configure and fine tune it for Check Point R75.

Supported Versions

- R70
- R71
- R75

Supported Operating Systems

- SecurePlatform 2.4/2.6
- IPSO
- CrossBeam

Supported Appliances

Multi-core processor platforms.

Before You Start

Related Documentation and Assumed Knowledge

- How connection table limit capacity behaves in CoreXL - sk35990 (https://supportcenter.checkpoint.com/supportcenter/portal?eventSubmit_doGoviewsolutiondetails=&solutionid=sk35990&js_peid=P-114a7bc3b09-10006&partition=Internal&product=CoreXL”)
- CoreXL Known Limitations - sk61701 (https://supportcenter.checkpoint.com/supportcenter/portal?eventSubmit_doGoviewsolutiondetails=&solutionid=sk61701&js_peid=P-114a7bc3b09-10006&partition=General&product=CoreXL”)
- Advanced system administration of Check Point products.
- Standalone deployment is not supported.
- In CoreXL, the number of concurrent connections is the sum of concurrent connections that exist in all instances. This value is then compared to the maximum value.
- CoreXL does not support Check Point Suite with these features:
  - Check Point QoS (Quality of Service)
  - Route-Based VPN
  - IP Pool NAT
To enable a non-supported feature, disable CoreXL with `cpconfig`, and reboot the gateway.

### Impact on the Environment and Warnings
In a CoreXL gateway, the firewall kernel is replicated multiple times.

### Configuring and Tuning CoreXL on SecurePlatform

CoreXL is a technology that enhances the performance of Security Gateways on multi-core processor platforms. CoreXL allows the cores to perform multiple tasks concurrently to enhance the gateway performance levels.

This feature provides scalability of performance, according to the number of processor cores on a single machine. No change to network topology or management is required. CoreXL joins ClusterXL Load Sharing and SecureXL as part of the Check Point traffic acceleration technologies.

The firewall kernel is replicated a number of times in a CoreXL gateway. Each instance or replicated copy of the firewall kernel runs on one processor core. These instances handle traffic concurrently, and each instance is a complete and independent inspection kernel.

A CoreXL gateway functions as a regular Security Gateway, with regards to a management configuration, network topology, and security policies. The kernel instances of a gateway handle traffic that goes through the same gateway interfaces, and apply the same gateway security policy.

The number of kernel instances is derived from the total amount of cores on the system:

<table>
<thead>
<tr>
<th>Number of cores</th>
<th>Number of Kernel Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CoreXL is disabled</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>More than 8</td>
<td>Number of cores, minus 4</td>
</tr>
</tbody>
</table>

When Performance Pack is installed, the default affinity setting for all interfaces is **Automatic**. Traffic from all interfaces is directed to the core that processes the Secure Network Distributor (SND).

### Begin Configuration

**To Enable or Disable CoreXL Configuration:**

1. On the gateway, run `cpconfig`
2. Enter 7 to select **Configure Check Point CoreXL**.

```
Configuration Options:
-----------------------
(1) Licenses and contracts
(2) SNMP Extension
(3) FRCS#11 Token
(4) Random Pool
(5) Secure Internal Communication
(6) Enable cluster membership for this gateway
(7) Configure Check Point CoreXL
(8) Automatic start of Check Point Products
(9) Exit

Enter your choice {1-9} : 7
```

3. Enable CoreXL.
4. Reboot the gateway.

**To Configure the Number of Instances:**
1. When CoreXL is enabled, enter 1 to select **Change the number of firewall instances**.

```
Configuring Configure Check Point CoreXL...

CoreXL is currently enabled with 2 firewall instances.

(1) Change the number of firewall instances
(2) Disable Check Point CoreXL
(3) Exit

Enter your choice {1-3} : 1

This machine has 3 CPUs.

How many firewall instances would you like to enable [2 to 3] [2] ?
```

2. Reboot the gateway.

**Allocating Processor Cores**

The CoreXL software architecture includes the Secure Network Distributor (SND) that is responsible for:

- The process of traffic that comes from the network interfaces.
- The secure acceleration of authorized packets (when Performance Pack is used).
- The Distribution of non-accelerated packets among kernel instances.

Traffic that enters network interface cards (NICs) is directed to a core that processes the SND. The association of a particular interface with a processor core is called the Interface Affinity with that core. This affinity causes the interface traffic to be directed to that core and the SND to process on that core. To set a kernel instance or a process for a particular core is referred to as the affinity of the instance or process with that core.

The default affinity setting for all interfaces is Automatic. Automatic affinity means that if Performance Pack is used, the affinity for each interface is automatically reset every 60 seconds and balanced between available cores. When Performance Pack is not used, the default affinities of all interfaces are to one available core. In both cases, any core that processes a kernel instance, or defined as the affinity for another process, is considered unavailable and is not set as the affinity for any interface.
In some cases, it may be advisable to change the distribution of kernel instances, the SND, and other processes, among the processor cores. To do so, the affinities of different NICs (interfaces) and/or processes need to be changed. However, to ensure CoreXL efficiency, all interface traffic must be directed to cores that do not process kernel instances. Therefore, if you change affinities of interfaces or other processes, you need to set the number of kernel instances accordingly, and ensure that the instances are processed on other cores.

Under normal circumstances, it is not recommended for the SND and an instance to share a core. However, it is necessary when the machine used has exactly two cores.

To Add Processor Cores to the Hardware:

More processor cores on the hardware platform do not automatically mean more kernel instances. If the number of kernel instances is not increased, CoreXL does not utilize some of the processor cores. When the hardware is upgraded, you can reconfigure the number of kernel instances.

1. To reconfigure the number of kernel instances, run: `cpconfig`
2. To implement the change in number of kernel instances, re-install the gateway (also when upgraded manually).

To Check if the SND Slows the Traffic Down:

When the SND slows the traffic down, and your platform contains enough cores to reduce the number of kernel instances, you may want to allocate an additional core to the SND.

1. To identify the processor core to which the interfaces direct traffic, run `fw ctl affinity -l -r`

   ![Example output of `fw ctl affinity -l -r` command]

   ```
   [Expert@tamdcorexlr75mu]# fw ctl affinity -l -r
   CPU 0: eth0
   eth1
   CPU 1: eth0
   CPU 2: eth1
   ALL: mpsdemon fundamentals
   [Expert@tamdcorexlr75mu]#
   ```

2. On the CoreXL gateway, run the `top` command, and check the values for the different cores under the `idle` column (in the `Cpu(s)` line. In this Example: 99.9%id)

   ![Example output of `top` command]

   ```
   top - 15:00:10 up 33 min, 1 user, load average: 0.00, 0.00, 0.00
   Tasks: 75 total, 1 running, 74 sleeping, 0 stopped, 0 zombie
   Cpu(s): 0.0us, 0.0%sy, 0.0%ni, 99.9%id, 0.1%wa, 0.0%ki, 0.0%si, 0.0%st
   Mem: 1014312k total, 273556k used, 735756k free, 14888k buffers
   Swap: 2128604k total, 0k used, 2128604k free, 0k cached
   PID USER      PR  NI VIRT  RES  SHR   %CPU   %MEM    TIME+ COMMAND
   4094 root     20  0   2048  994  784    0.1    0.00   0:00.04   top
   1 root      15  0   1600   504  452    0.0    0.00   0:00.66   init
   2 root      15  0   2048  994  784    0.0    0.00   0:00.00   migration
   3 root      15  0   2048  994  784    0.0    0.00   0:00.00   migration
   4 root      15  0   2048  994  784    0.0    0.00   0:00.00   migration
   5 root      15  0   2048  994  784    0.0    0.00   0:00.00   migration
   6 root      15  0   2048  994  784    0.0    0.00   0:00.00   migration
   7 root      15  0   2048  994  784    0.0    0.00   0:00.00   migration
   8 root      15  0   2048  994  784    0.0    0.00   0:00.00   migration
   9 root      15  0   2048  994  784    0.0    0.00   0:00.00   migration
   10 root     15  0   2048  994  784    0.0    0.00   0:00.00   migration
   11 root      15  0   2048  994  784    0.0    0.00   0:00.00   migration
   ```
To Allocate an Additional Processor Core to the SND:

**Note** - It is not recommended to allocate an additional core to the SND unless all these conditions are met:

- Your platform has at least eight processor cores.
- The idle value for the core, in which the SND is currently processed, is in the 0%-5% range.
- The sum of the idle values for the cores that process kernel instances is significantly higher than 100%.

1. To reduce the number of kernel instances, run: `cpconfig`.
2. Set interface affinities to the remainder cores:

   **With Performance Pack:**

   In Automatic mode (default setting of the Performance Pack sim affinity command), interface affinities are automatically distributed among cores that do not process kernel instances, and are not set as the affinity for any other process.

   **Without Performance Pack:**

   Interface affinities are loaded at boot from a configuration text file called `fwaffinity.conf` that is located under `$FWDIR/conf`.

   In the text file, lines that begin with `i`, define interface affinities.

   ```bash
   [Expert@tancorexlr75gw]# more fwaffinity.conf
   # Process / Interface Affinity Settings
   # -------------------------------------
   # Each line should contain:
   # 1. A type - 1 character. "i" for interface, "n" for process name, "k" for kernel instance.
   # 2. An ID - interface name, process name, or kernel instance number.
   # a. For interfaces, you can also write "default", and the setting would apply to any interface not mentioned in the file.
   # 3. The desired affinity. Either:
   # a. One or more CPU numbers.
   # b. "all" - all CPUs are eligible.
   # c. "ignore" - do nothing for this entry.
   # d. "auto" - use any free CPU. A free CPU is one that doesn't appear in any line in this file, and doesn't run a worker thread.
   #
   # i default auto
   [Expert@tancorexlr75gw]#
   ```

   **To allocate one processor core to the SND:**

   Make sure that `fwaffinity.conf` contains this line: `i default auto`, and that it contains no other lines that begin with `i` (that would define explicit interface affinities).

   **To allocate two processor cores to the SND:** set explicit interface affinities to the other cores:

   **Note** - In the case of multiple interfaces, you need to set each interface for each core. Try to achieve a balance of expected traffic between the cores. You can later check the balance with the `top` command.

   a) To set the affinity for each interface, edit `fwaffinity.conf` to contain one line that begins with `i` for each interface. Each of these lines should follow this syntax:

   ```bash
   i <interfacename> <cpuid>
   ```

   **Where:**

   Instead of `<interfacename>`, enter the interface name.

   Instead of `<cpuid>`, enter the number of processor core to be set as the affinity of that interface.
For example: if you want the traffic from eth0 and eth1 to go to core #0, and the traffic from eth2 to
go to core #1, enter these lines in `fwaffinity.conf`:

```
i eth0 0
i eth1 0
i eth2 1
```

**OR**

Define explicit interface affinities for only one processor core, and define the other core as the
default affinity for the other interfaces. To do so, enter `default` instead of `<interface name>`.

For example: if you want the traffic from eth0 and eth1 to go to core #0, and the traffic from eth2 to
go to core #1, enter these lines in `fwaffinity.conf`:

```
i eth2 1
i default 0
```

b) For the `fwaffinity.conf` settings to take effect, run `$FWDIR/scripts/fwaffinity_apply`.

*Note* - The affinity of virtual interfaces can also be set with their physical
interface(s).

3. Reboot to implement the new configuration.

**To Allocate a Core for Heavy Logging:**

If the gateway performs heavy logging, you can allocate a processor core to the fwd daemon, which
performs the logging, to reduce the number of cores available for kernel instances.

1. To reduce the number of kernel instances, run: `cpconfig`

2. To set the fwd daemon affinity:

a) Check which cores process the kernel instances, and which cores handle interface traffic with:
   `fw ctl affinity -l -r`

b) Set the fwd daemon affinity to the remainder core to allocate that core to it.

   *Note* - It is important to avoid the core(s) that process the SND only if these cores are
defined as explicit affinities of interfaces. If interface affinities are set to Automatic, any
core that does not process a kernel instance can be used for the fwd daemon, and
interface traffic is automatically diverted to other cores.

   (i) Edit the `fwaffinity.conf` configuration text file that is located in `$FWDIR/conf`. Add: `fwd
   <cpuid>`

   Where: instead of `<cpuid>`, enter the number of the processor core to be set as the affinity of the
fwd daemon.

   For example: to set core #2 as the affinity of the fwd daemon, add:
   `n fwd 2`

   (ii) Reboot for the `fwaffinity.conf` settings to take effect.

**Using the Command Line Reference**

Affinity settings are controlled by the `fwaffinity_apply` script file, which executes automatically at boot.
When you make a change to affinity settings, the settings do not take effect until you either manually
execute the `fwaffinity_apply` script, or reboot.

`fwaffinity_apply` executes affinity definitions according to the information in the `fwaffinity.conf`
text file. To change affinity settings, edit the text file.

*Note* - When Performance Pack is used, interface affinities are only defined by the
Performance Pack `sim affinity` command. The `fwaffinity.conf` interface affinity
settings are ignored.
To Edit fwaffinity.conf:

fwaffinity.conf is located in the $FWDIR/conf directory. Its parameters are:

- `<type>`
  - i interface
  - n Check Point daemon
  - k kernel instance
- `<id>`
  - interface name if `<type>` = i
  - daemon name if `<type>` = n
  - instance number if `<type>` = k
- `default` interfaces that are not specified in another line.
- `<cpuid>`
- `<number>` number(s) of processor core(s) to be set as the affinity.
- `all` all processor cores are available to the interface traffic, daemon, or kernel instance.
- `ignore` no specified affinity (useful for exclusion of an interface from a default setting).
- `auto` Automatic mode.

**Note** - Interfaces that share an IRQ cannot have different cores as their affinities. That includes the instance where one interface is set as default affinity. You must either set both interfaces to the same affinity, or use `ignore` for one of them. To view the IRQs of all interfaces, run: `fw ctl affinity -l -v -a` and `fwaffinity_apply`

In the $FWDIR/scripts directory, in `fwaffinity_apply`, run: $FWDIR/scripts/fwaffinity_apply <option>

Where instead of `<option>`, enter:

- q for Quiet mode, to print only error messages.
- t for `<type>`, to only apply affinity for the specified type.
- f to set interface affinity even if automatic affinity is active.

To Set Affinities:

The `fw ctl affinity` command controls affinity settings. However, `fw ctl affinity` settings do not persist through a restart of the Security Gateway.

- To set affinities, run: `fw ctl affinity -s`
- To list affinities, run: `fw ctl affinity -l`

To Check Kernel Information:

The `fw ctl multik stat` (multi-kernel statistics) command displays information for each kernel instance. The state and processor core number of each instance is displayed with:

- The number of connections currently being handled.
- The peak number of concurrent connections the instance ever handled.

```bash
[Expert@standbcorexl275gv]# fw ctl multik stat
ID | Active | CPU | Connections | Peak
-----------------------------------------------
 0 |  Yes  |  2  |     0       |   0
 1 |  Yes  |  1  |     0       |   0
 2 |  Yes  |  0  |     0       |   0
[Expert@standbcorexl275gv]#
```
Completing the Procedure

- Use the given commands to fine tune CoreXL to your system requirements.
- Make sure the policy is pushed and traffic passes through the firewall.

Verifying the Procedure

- Verify CoreXL configuration with the commands given in this document.
- Run `fw ctl multik stat` to display information for each kernel instance.
- Run `fw ctl affinity -l -r` to identify the processor core to which the interfaces direct traffic.
- Verify that the configuration is correct in `$FWDIR/conf/fwaffinity.conf`
Index

A
Allocating Processor Cores • 7

B
Before You Start • 5
Begin Configuration • 6

C
Completing the Procedure • 12
Configuring and Tuning CoreXL on SecurePlatform • 6

H
How To Configure and Tune CoreXL on SecurePlatform • 5

I
Impact on the Environment and Warnings • 6
Important Information • 3

O
Objective • 5

R
Related Documentation and Assumed Knowledge • 5

S
Supported Appliances • 5
Supported Operating Systems • 5
Supported Versions • 5

U
Using the Command Line Reference • 10

V
Verifying the Procedure • 12