

如何在 CentOS5 上安裝 Dazuko

目的

安裝 Dazuko 模組在 Centos Linux 5 的系統上。Dazuko 提供一個 Virtual Device Driver.給 Eset 作為即時的文件存檔病毒掃描。換句話說，安裝了 Dazuko 模組後，Eset 就可以在 OS Level 進行病毒掃描。

準備工作

1. 找出現有的內核(kernel)版本和使用中的處理器 (如 i686)，方法如下：

```
[barry@localhost ~]$ uname -a  
Linux localhost.localdomain 2.6.18-53.el5 #1 SMP Mon Nov 12 02:22:48 EST  
2007 i686 i686 i386 GNU/Linux
```

即這系統現時運行的內核是 2.6.28-53-53 版本，處理器是 **i686**，並運行 SMP mode.

2. 下載適合這版本 CentOS Linux 有關的最新內核原始 Source Code RPM. 例如，CentOS 5 的最新 Source Kernel RPM 可以在：
<http://mirror.centos.org/centos/5/updates/SRPMS/kernel-2.6.18-92.1.6.el5.src.rpm>
找到。
3. 下載最新的 dazuko 原始檔。現時最新的版本是 **dazuko-2.3.5.tar.gz**
4. 由於編譯 (compile) 核心需要大量編譯軟件和檔案。因此最佳的做法是安裝 “**Devlopment Tools**” 套件。這可以由安裝光碟找到。也可一同安裝 “Development Libraries” 套件。



編譯核心

所有 Red Hat/CentOS 的 Linux 核心都將 “Linux Default Capabilities security” 模組編譯在內核中。但 Dazuko 需要將這部份分開，先上了 Dazuko 模組，後才加上 “Linux Default Capabilities security” 模組。

由於原來的 Source RPM 是基於 Red Hat Linux 的，而現在是在 CentOS Linux Compile，故有少少技巧：

1. 安裝 Source Kernel RPM

```
rpm -ivh kernel-2.6.8-92.1.6.el5.src.rpm
```

2. `cd /usr/src/redhat/SPECS` 目錄

A. 由於我們只需要 i686 的 kernel，因此以下的設定

```
%define with_smp      %{?_without_smp:      0}
%{?!_without_smp:    1}
# kernel-PAE (only valid for i686)
%define with_pae      %{?_without_pae:      0}
%{?!_without_pae:    1}
```

```

# kernel-xen (only valid for i686, x86_64 and ia64)
#define with_xen      % {?_without_xen:      0}
% {?!_without_xen:   1}
# kernel-kdump (only valid for ppc64)
#define with_kdump    % {?_without_kdump:    0}
% {?!_without_kdump: 1}
# kernel-debug
#define with_debug    % {?_without_debug:    0}
% {!?!_without_debug: 1}
# kernel-doc
#define with_doc      % {?_without_doc:      0}
% {?!_without_doc:   1}
# kernel-headers
#define with_headers  % {?_without_headers:  0}
% {?!_without_headers: 1}
# kernel-debuginfo
#define with_debuginfo % {?_without_debuginfo: 0}
% {!?!_without_debuginfo: 1}

#define with_kabichk  % {?_without_kabichk:  0}
% {?!_without_kabichk: 1}

```

全改爲

```

#define with_smp      0
# kernel-PAE (only valid for i686)
#define with_pae      0
# kernel-xen (only valid for i686, x86_64 and ia64)
#define with_xen      0
# kernel-kdump (only valid for ppc64)
#define with_kdump    0
# kernel-debug
#define with_debug    0
# kernel-doc
#define with_doc      0
# kernel-headers
#define with_headers  0
# kernel-debuginfo
#define with_debuginfo 0

```

```
%define with_kabichk 0
```

B. 設定 buildid

```
%define buildid .dazuko
```

3. `cd /usr/src/redhat/SOURCES` 目錄

根據你選的處理器，選擇合適的 `.config` 檔。例如現在我們是 `compile i686` 的，我們就在此目錄內找出

```
kernel-2.6.18-i686.config
```

這檔來更改，將原來

```
CONFIG_SECURITY_CAPABILITIES=y
```

改爲

```
CONFIG_SECURITY_CAPABILITIES=m
```

4. 轉回 `/usr/src/redhat/SPECS` 目錄

開始 `compile kernel`

```
rpmbuild -bb --target=i686 kernel-2.6.spec.dazuko 2>/tmp/kernel-err.log
```

5. 編譯時間大約需要半小時（決定於閣下機器的速度）。完成後的 `binary RPM` 是放在 `/usr/src/redhat/RPMS/i686` 這目錄下。

```
rpm -ivh kernel-2.6.18-92.1.6.el5.dazuko.i686.rpm
```

就可安將一個沒有“Security Capabilities” build in 的新內核。

6. 檢視 `/etc/grub.conf` 這檔，肯定下次開機時會自動選用新的內核開機。

7. 重新開機。

編譯 dazuko 內核模組

1. 解壓原始檔 (`dazuko-2.3.5.tar.gz`)

```
tar xvzf dazuko-2.3.5.tar.gz
```

因為較新的 Kernel 有 LSM API 問題 (詳情可看 <http://www.dazuko.org/tgen.shtml>)，故此 configure 時比較複雜。

2. cd dazuko-2.3.5

```
[root@localhost dazuko-2.3.5]# ./configure --enable-syscalls
--mapfile=/boot/System.map-2.6.18-92.1.6.el5.dazuko --sct-readonly
--kernelsrcdir=/usr/src/redhat/BUILD/kernel-2.6.18/linux-2.6.18.i686/
checking host system type... Linux
checking for make utility... ok (make)
checking for C compiler... ok (cc)
kernel build source in /usr/src/redhat/BUILD/kernel-2.6.18/linux-2.6.18.i686... yes
acquiring Linux kernel code configuration... ok
checking if Linux is RSBAC patched... no
checking if devfs is enabled... no
discovered host system... Linux (2.6.18)
checking whether __d_path() is exported... yes
checking for System.map file... ok (/boot/System.map-2.6.18-92.1.6.el5.dazuko)
locating sys_call_table... ok (0xc060e4e0)
checking sys_call_table status... read-only (forced)
locating do_execve... ok (0xc047ad9f)
identifying device API... ok
inspecting class type... ok (class)
inspecting suspend function... ok (suspend2)
inspecting task_struct structure... ok (using parent)
configure: creating Makefile
configure: creating library/Makefile
configure: creating example_c/Makefile

./configure successful

=====
Configuration summary
=====

module events = ON_OPEN ON_CLOSE ON_EXEC
devfs support = no
rsbac support = no
```

```
hooking via syscalls = yes
local __d_path() = no
path resolution = registered daemon context
module debug = no
library 1.x compatibility = yes
```

3. 編譯及安裝

make; make install

4. 試驗 dazuko 模組

```
[root@localhost ~]# modprobe dazuko
[root@localhost ~]# lsmod | grep dazuko
dazuko                42364  0
```

5. 安裝完成