Redbooks Paper

Rob van der Heij

Running Linux Guests with less than CP Class G Privilege

Introduction

When a system is connected to the Internet, handed over to customers, or otherwise placed in an uncontrolled environment, there is a risk that the server may get compromised. Linux servers are no exception to this. Proper security policies, internet firewalls, and packet filtering all act to minimize this risk and impact. When running Linux servers as z/VM guests, we also need to consider interactions between the Linux virtual machines and the z/VM host. As this type of interaction is not limited to network connectivity, firewalls are less effective in controlling it.

With z/VM, virtual machines can be isolated the same way as Logical Partitions (LPARs) are isolated on a zSeries machine. However, in most cases you will want to make the isolation less perfect in order to share resources, manage the virtual machines, and allow communication between virtual machines. Historically, privileges granted to a virtual machine matched running CMS applications inside a corporate network. In this environment, it is desirable to lower the fences between virtual machines. The risk in this situation is mitigated by an acceptable use policy where employees risk their job for attempting to acquire higher privileges than authorized.

With virtual machines running Linux servers in a hostile environment, we need to review virtual machine privileges using different criteria. Knowing that unauthorized people may get hold of the root account in a Linux server, you will want to raise the fences between the virtual machines higher than what normally is done for CMS virtual machines. Because Linux virtual machines have less ability to share resources on z/VM, you fortunately can raise the fences between Linux virtual machines much higher without loss of function.

Scope of this paper

The purpose of this paper is to provide guidance in hardening a z/VM system to run Linux virtual machines in a hostile environment. The issues identified in this paper for a default z/VM configuration are not real exploits, in that a compromised Linux server could get unauthorized access to resources. Instead, most issues represent a potential Denial of

Service (DoS) where in the worst case, a compromised server can prevent other virtual machines access to resources they are authorized to use.

Because various systems management tasks in z/VM also run in virtual machines, such a DoS attack could cause problems for those systems management tasks. Not being able to run those tasks could make the system vulnerable for attacks. When multiple virtual machines are compromised, the networking facilities in Linux make it possible for a hacker to engage multiple virtual machines in such an attack. This means we also need to protect the system against a Distributed Denial of Service attach (DDoS). While the system may be immune for the bad behavior of a single virtual machine, we could imagine scenarios where a lot of virtual machines misbehaving in the same way at the same time cause a problem.

The guidelines in this paper allow you to run Linux virtual machines in a safer way. This does not necessarily mean that you are required to implement all recommendations, or that you must run all Linux virtual machines that way. If you are confident enough that for some Linux servers the risks are less, then you may decide to lower the fences for those Linux virtual machines. As always, you must make the trade-off between security, flexibility, and ease of use. This is best done with a good understanding of the requirements of your installation.

This paper takes a defensive approach for the design of such a hardened z/VM system, in that we close all facilities that are not strictly required to run Linux virtual machines. This is different from an approach where facilities are left open until an exposure is found. We believe the defensive approach of implementation is less complicated and easier to validate.

Another area of concern should be the functions that reveal information about the hosting environment or the other customers that have Linux servers on the same VM system. The information revealed this way may not be considered secret, but it can be used in an indirect way. Lists of server names or IP addresses can be used for systematic attacks (some installations used to disable **`Q NAMES`** for that reason). Various commands can also reveal *business sensitive* information about other customers (the number of servers used, the amount of disk space used for these servers), and the hosting environment (total capacity and utilization of the machine). Because it is hard to decide which information is business sensitive, the defensive approach is attractive to address those issues as well.

Interaction with CP

The Control Program (CP) is responsible for executing privileged operations on behalf of a virtual machine, and does this in such a way that virtual machines are isolated from each other where necessary. CP is also responsible for the configuration of virtual machines and for maintaining the control structures used by hardware protection mechanisms used for operating virtual machines. Various interfaces allow the virtual machine to request services from CP. When a virtual machine could trick CP into doing things that should not be done, that virtual machine could impact the integrity of the system.

This section discusses CP Commands and Diagnose Codes necessary to run Linux, and describes how to set up a CP privilege class definition that prevents abuse of these interfaces.

CP commands

Probably the most obvious way for the virtual machine to access its environment is through the CP commands that allow the virtual machine to interact with CP, the z/VM hypervisor.

Authorization for CP commands is based on the *privilege class* concept. The CP commands are assigned a privilege class based on the function they perform (such as, system operation,

spool processing, real device operation, general user). Through their entry in the CP Directory, users are assigned one or more privilege classes based on the tasks they need to perform (primary operator, spool operator, I/O operator, performance specialist, end-user). The CP command processor will only allow a user to invoke a CP command when the user has the proper privilege class for the command.

VM traditionally used seven privilege classes, A-G (where G is the general user). Current z/VM releases allow installations to implement a more granular access control through changes to privilege class assignment with MODIFY statements in the CP System Configuration file. [1]

A virtual machine can issue CP commands through a special machine instruction (the Diagnose 8 instruction) that is intercepted by CP. Some Linux distributions contain the cpint package, which comes with the **hcp** command to allow the root user to issue CP commands this way. Note that the **hcp** command is just a convenient way to issue these commands. The root user could write custom code in the Linux server to achieve the same effect. This means no additional protection is achieved by hiding or modifying the **hcp** command.

The default privilege class G is typically used for general CMS users. The end-users on such a system will be assigned class G, and users with special tasks get class G plus one or more other classes. As explained above, this default class G is more powerful than necessary for Linux virtual machines. Since taking commands away from class G would impact existing CMS users, we need to create a new privilege class used for Linux virtual machines. In this paper we refer to this new privilege class as " L" but any available privilege class could be used for it (or even multiple ones when different levels of less privileged virtual machines are required). The Linux virtual machines will get that class L through their CP directory entry.

The defensive approach followed in this paper dictates that we start with an "empty" privilege class L and add commands as the need arises, rather than copy class G and take out some things that are potentially harmful.

Note: Some CP commands come in different versions depending on the privilege class of the user that issues the command. These versions may have different function and accept different parameters. For example, when executed by a class G user the CP **QUERY** command refers to virtual devices; when executed by class B users, **QUERY** refers to real devices by default.

CP commands used by Linux

Study of the Linux source code and the use of CP **TRACE** facilities reveals which CP commands are used by the Linux kernel and kernel modules. This section discusses these commands and explains why it is safe to add this command to privilege class L for Linux virtual machines.

Commands used by the current distributions

The following CP commands are used by the kernel and common device drivers of current (early 2004) commercial Linux distributions:

QUERY TERMINAL	It is unclear why the kernel issues this command (it does not use the result to decide whether to issue the **TERMINAL** command to change terminal settings). We do not believe this command poses a security exposure.
TERMINAL AUTOCR OFF	This allows the console driver to compose output lines on the linemode display.

[1] Earlier VM releases implemented this using the User Class Restructure facility.

TERMINAL CONMODE 3215 The **TERMINAL** command is used to set 3215 linemode rather than 3270. Since the default is already 3215 mode this is not necessary. Unfortunately z/VM does not allow us to remove authorization to this command and still keep authorization to the **TERMINAL AUTOCR** command. This means the virtual machine is also able to set **CONMODE** to 3270. A virtual console set to CONMODE 3270 would make it almost impossible for support staff to logon to the virtual machine and issue commands on the console.[2]

IPL The **IPL** command is only used in reboot of Linux. The initial IPL during logon as stated in the CP directory does not require the virtual machine to execute the **IPL** command. If we allow the Linux virtual machine to use the **IPL** command, it can also IPL from another NSS or another device than specified in the CP directory.

SET PAGEX ON Only used by the kernel when PFAULT is not available (that is, before z/VM 4.1).

Commands used by recent patches

The patches on the IBM developerWorks Web site allow Linux to exploit more VM function. Future distributions will most likely contain this code as well.

SET SMSG IUCV This command is used by Collaborative Memory Management (supplied with the 2.4.21 kernel, patch 10). It allows the CMM driver to set up an SMSG handler to receive instructions to change Linux memory management strategy.

When a Linux virtual machine can issue the **SET SMSG IUCV** command to enable the CMM driver to catch requests, it can also issue the **SET SMSG IUCV OFF** command (causing the virtual machine to ignore requests and thus remove itself from the control of external system management tools). This does not create an additional exposure because Linux root access is required to issue the command (and root has other ways to disable CMM).

RECORDING EREP The vmlogrdr kernel module (with 2.4.21 kernel, patch 10) issues this command (necessary when a Linux virtual machine is configured to receive EREP data). If you want a Linux guest to receive EREP data, use a virtual machine managed only by system staff.

Note: These CP commands are typically only used during the boot process of the system, or when the particular device driver is loaded.

CP class ANY

A complication with setting up the privilege class L is that some CP commands are assigned to privilege class ANY, which means they are not protected by any CP privilege class. The main reason for this is that these commands must be used before a userid is logged on. At that time there is no privilege class is involved yet (the **LOGON** command is an obvious example). Some commands have been assigned class ANY because they were considered harmless. Because these are available to any privilege class, a class L virtual machine can also issue these commands. To prevent this, we must change the privilege class assignment for these commands from ANY to G, so that CMS users are still be able to use the command.

[2] A sufficiently authorized support person could issue a CP **SEND** command against the virtual machine and set it back to 3215 to recover.

Removing commands from class ANY does create some problems for users with no privilege class (pre-logon and users who used **SET PRIVCLASS** incorrectly). This means we must deviate here from the defensive approach and restrict only the commands that are dangerous.

The following section lists the class ANY commands, which could be exploited by a compromised Linux virtual machine and should be candidates for privilege class reassignment.

DISCONNECT This could be inconvenient when system staff personnel are logged on to the virtual machine and a hacker issues a **DISCONNECT** command through the diagnose 8 interface. Note that when the Linux virtual machine can not issue the **DISCONNECT** command, system support staff logged on to the virtual machine can not do so either (but they can use a **FORCE DISC** from a higher privileged userid for that).

LOGOFF This command is similar to **DISCONNECT**, but more disruptive because the re-IPL of the virtual machine could be time consuming, since the file systems are not cleanly unmounted when the virtual machine is logged off. If the **LOGOFF** is made class G it should not cause problems for existing users, except when they disabled their own class G privilege.

MESSAGE The **MESSAGE** command is potentially dangerous when used against automated services that accept commands this way. Even when the service is properly hardened to refuse commands from unauthorized users, an excessive amount of messages could prevent it from doing its real work. The **MESSAGE** command may also be used by a hacker to learn which users are logged on. When class G is required for the MESSAGE command it can not be used before logon, but this is only a minor loss in function[3].

SLEEP This command makes the Linux server unresponsive for some time. Clearly someone with root access can do other things that prevent authorized users to access the system, so **SLEEP** is only a very minor risk. When MESSAGE is not class ANY there is no need for the SLEEP command either[4].

Note: Installations that use RACF/VM as the External Security Manager can use a special feature of RACF/VM to prevent the use of some of the “pre-logon” commands (like **MESSAGE**). For installations that require those commands to be disabled before logon, this feature is an alternative for privilege class assignment of those commands. However, this is **not** effective for the restriction of the **MESSAGE** command that we describe here. Because a running Linux virtual machine is not in the pre-logon state, it would not prevent the running Linux server from issuing **MESSAGE** commands. So even when this RACF feature is used, the change in privilege class is still required. RACF/VM does not provide a way to control the **MESSAGE** command outside the pre-logon state.

[3] Traditionally the **MESSAGE** command was allowed before logon to allow a conversation with the operator or with other users. In those days the real device address of the terminal was often enough to verify the identity of the person sending the messages. With the introduction of virtual terminals through VTAM and TCP/IP it is harder to determine who is sending the messages, and many installations have chosen to disable the command before logon.

[4] The reason for allowing the **SLEEP** command before logon is to be able to take the terminal out of CP READ while waiting for a response from another user (since messages will not be displayed while the terminal is in CP READ).

The following CP commands only reveal information about the virtual machine or other virtual machines. Since these are not used by Linux, it would be wise to remove them from ANY and require privilege class G instead.

COMMANDS This only shows the available commands and could encourage an intruder to explore the possibilities (but the list has become very short for class L now).

QUERY BYUSER This command shows which user logged on the Linux virtual machine with **LOGON BY** (if any). The concept of logging on to the virtual machine console does not make much sense to Linux, so it would only reveal the name of an authorized user.

QUERY COMMANDS This is identical to **COMMANDS**.

QUERY PRIVCLASS When Linux is not supposed to modify its own privilege class, it does not need this information either.

QUERY USERS The number of logged on virtual machines is not valuable information for a Linux server.

SET PRIVCLASS This command is class ANY so that a user will always be able to get back to the privilege class assigned in the directory. If the CP directory entry for the Linux virtual machine restricts it to the proper privilege class, the command does little harm but is not needed either. When the two-phase approach is used to start the Linux virtual machine as described in “An alternative approach” on page 13, it is important to restrict this command.

SILENTLY The command works as a “wrapper” around a limited set of CP commands to suppress messages. Currently none of these CP commands is used by Linux, so the **SILENTLY** command is not needed either.

The following CP commands do not work through the diagnose 8 interface, but can only be issued from the virtual machine console. There is no need to restrict them, and doing so would even cause trouble on the logon screen.

DIAL

LOGON

UNDIAL

The list of commands available to class ANY can be obtained using the commands shown in Example 1 (executed from a privileged userid). The alternative to issue **COMMANDS** from the userid itself does not work when that particular command is already disabled.

Example 1 Listing commands available to class ANY

```
pipe cp query cpcmd * | joincont leading / / x15 | locate ,<ANY>, | split 15 | cons
pipe cp query cpcmd query subc * | joincont leading / / x15 | locate ,<ANY>, | split 15 | cons
pipe cp query cpcmd set subc * | joincont leading / / x15 | locate ,<ANY>, | split 15 | cons
```

Compatibility

Existing z/VM installations may already have modified CP privilege class assignment to satisfy their security needs (the IBM internal systems are an obvious example). In such a situation it may not be possible to use the set of MODIFY statements provided in “Sample MODIFY statements for the default z/VM 4.4 installation” on page 13. To define the privilege class for Linux virtual machines in such an environment, the systems programmer should review the existing privilege class assignments and merge the privilege class assignments

suggested here into the existing configuration. This also applies to installations that have defined additional CP commands.

Important: Earlier VM releases implemented changes in CP privilege class by means of special spool files created by the CMS **OVERRIDE** command. Installations still using this feature should be aware that adding a MODIFY statement to the System Configuration file will disable the existing UCR file. Those installations must either implement these changes in their existing UCR file or convert their existing privilege classes assignments using a MODIFY statements.

Diagnose codes

Access control for the IBM supplied CP Diagnose Codes is done with the same privilege class structure as is used for CP commands. We find that most diagnose codes have been assigned to class ANY. This is not because they would be used before logon, but because CP decided not to restrict access. The table lists the diagnose codes currently used by Linux. The defensive approach we follow implies that we want to move all remaining codes from ANY to G to make sure that the Linux machines in class L do not get this for free. This does not cause problems for existing users that have class G.

Diagnose codes used by the Linux kernel

The following IBM defined diagnose codes used by the Linux kernel and kernel modules as currently provided by the commercial distributions.

Diag 08 This is required to issue CP commands. Access to the individual CP commands is done with privilege class.

Diag 44 This is issued by CP to voluntarily end the timeslice.

Diag 60 This will get the virtual storage size.

Diag 210 This will obtain device information from CP

Diag 214 Pending page release. This diagnose code is not used by Linux but for some reason CP does not allow us to take it away from ANY

Diag 250 CP Block I/O, used by the dasd_diag_mod.

Diag 258 PFAULT macro and related undocumented function. Linux currently uses this to enable Pseudo Page Fault support.

Recent patches from the IBM developerWorks Web site also use the following diagnose codes:

Diag 10 Used by the Collaborative Memory Management (supplied with 2.4.21 kernel patch 10) to drop virtual machine pages and thus reduce the memory requirements of the virtual machine. This code is still very new and we still need to investigate the potential risks with this.

Diag 64 Allows Linux to attach a DCSS. This is used by the dcss block device driver to implement a block device in memory, and is used by the xip2 filesystem to share Linux binaries in DCSS among virtual machines. See also “Shared segments” on page 10.

Diag DC Used by the applmon driver (in the cpint package) to generate APPLMON monitor records. When these are being fed into some real measurement process you want to make sure that fake numbers are not going to confuse things. A compromised Linux server could create an excessive amount of account records causing damage to the system and impacting performance reporting processes (and hide the abuse inside the Linux server). The virtual

machine must also have an OPTION APPLMON directory statement to enable generation of monitor data, and monitoring for this user must be started with the **MONITOR** command (refer to the documentation of your performance monitor).

Communication between virtual machines

In addition to interaction with CP as discussed, the communication between virtual machines must be reviewed to assure that one virtual machine can not affect the integrity of another virtual machine. This is not restricted to communication between Linux virtual machines, but includes communication between a Linux virtual machine and virtual machines that are part of the z/VM infrastructure.

IUCV

Because IUCV is implemented with a pseudo instruction rather than a diagnose code, we can not restrict access to it through CP privilege class assignment. Fortunately, authorization for IUCV connections is controlled by statements in the CP directory. For a Linux server A to establish a connection with server B, it is necessary for user A to have an IUCV directory statement that lists B. The exceptions to this are designed to be used for special services:

IUCV ALLOW Allow any user in the system to establish a connection with this server

IUCV ANY Allow this user to establish a connection with any other user

Confusion about these options have caused some installations to apply the combination of IUCV ALLOW and IUCV ANY as the first attempt to solve an obscure problem[5].

It should be obvious we do not want to give IUCV ANY to a Linux machine that we do not fully control. A virtual machine defined with IUCV ALLOW is an easy target for connections from Linux virtual machines without explicit authorization. IUCV ALLOW is frequently used for system services such as SFS servers, AVSVM, TSAFVM, RACFVM, and TCP/IP related userids. While some of these applications have their own mechanism to reject unknown parties, a compromised Linux server could potentially block IUCV connections on such a server and thus cause a DoS. You should scan the CP directory for users with a an IUCV ALLOW statement and verify that an appropriate protection is implemented[6]. Whenever possible, IUCV ALLOW should be replaced with a list of authorized users. The disadvantage of this approach is that the server must be restarted to add userids to the access list.

Attention: Protection against abuse of IUCV by Linux virtual machines may require changes to the definition and setup of *other* virtual machines that are not related to Linux. You need to explain the changed environment to the owners of those services and have them agree to change the configuration.

In some cases, the Linux virtual machine must be able to establish an IUCV connection (for example, an IUCV point-to-point connection to the VM TCP/IP stack or a Linux virtual router). This requires an IUCV statement in the user directory of the Linux virtual machine. A compromised Linux server could potentially cause a DoS, by trying to establish many IUCV

[5] The default system delivered with DDR should define IUCV ALLOW and IUCV ANY for all TCP/IP related userids (whether they use IUCV or not). It should be possible to take out the IUCV statements for all TCP/IP related userids except for the IUCV ALLOW for the TCPIP virtual machine. A PendingConnectionLimit statement can be used to limit the number of half-open connections. If you want to be able to remove that last IUCV ALLOW as well, then the active TCP/IP related userids should have an IUCV TCPIP statement instead. We have not tested this approach for the full VM TCP/IP application suite.

[6] It would be nice if access control for IUCV were enhanced to allow connection by privilege class (for instance, all class G users) to exclude the Linux virtual machines with a single directory statement.

connections. To help prevent this, set MAXCONN for the Linux virtual machine not higher than strictly needed, and MAXCONN for the target virtual machine high enough so that a few compromised Linux servers can not lock others out. Applications that allow the Linux virtual machine to establish an IUCV connection must be configured to prevent abuse (for instance. a TCP/IP stack using an IUCV connection to a Linux server can use the PERMIT and RESTRICT statements to prevent use of other services).

Distributed IUCV

When the z/VM system is part of an ISFC collection and the System Configuration file enables Distributed IUCV, a virtual machine can establish an IUCV connection to targets in other systems in the collection. Distributed IUCV can be enabled or disabled on a system-wide basis only[7]. When distributed IUCV is disabled, the users on the system are not allowed to connect to resources on the other systems in the collection. When distributed IUCV is enabled on the z/VM system where the Linux virtual machines are running, the IUCV targets on other systems should be protected in the same way.

While the idea is that one should not connect systems with different administrative control, there may be some potential in connecting both internal and external systems with ISFC. This would for example allow systems management processes to work across Logical Partitions.

Important: APPC/VM connections use distributed IUCV as the transport mechanism, but the ability to connect to remote systems via APPC is not controlled by the Distributed IUCV setting in the System Configuration file. This means that all APPC resources in the CSE collection should be protected against unwanted connections from Linux. The local AVS gateway can protected with proper IUCV statements in the directory.

VMCF

CP does not provide a central mechanism for VMCF authorization as done for IUCV. This means it is up to the target virtual machine to reject connection requests. Existing services like TCP/IP, PVM, and the Performance Toolkit may have been configured with a friendly audience in mind. A VMCF conversation with the Performance Toolkit service virtual machine would provide detailed information about resource consumption of other virtual machines.

VMCF communication is done with a diagnose (Diag 68). Linux does not use VMCF so we can safely remove the diagnose code from privilege class ANY and thus keep it away from the Linux virtual machines.

Guest LAN and VSWITCH

With z/VM 4.4, Guest LAN (and Virtual Switch) have become the most obvious way to provide a Linux virtual machine with an IP connection. The Guest LAN can be defined either restricted or unrestricted. At first sight it may seem wise to define the Guest LAN as "restricted" so that virtual machines can only connect to the Guest LAN when specifically granted permission to do so. However, in “CP commands” on page 2 the **COUPLE** command was not listed as required by Linux virtual machine. This is possible because the directory statement to define virtual NIC can be made to automatically couple the virtual NIC to the Guest LAN. When the Linux virtual machine can not issue a **COUPLE** command, it can not connect to any other LAN than that arranged in the CP directory.

Although the restricted Guest LAN does not offer additional protection against virtual machines that can not issue the **COUPLE** command, it may still be attractive to define the Guest LAN as restricted. The explicit access list for the Guest LAN makes it easy to verify separation

[7] It would be very attractive if z/VM were enhanced to control cross-system IUCV connection by privilege class.

of user communities by someone without access to the CP Directory[8] and protects against mistakes by virtual machines that can issue the **COUPLE** command.

Note: If we allow a Linux virtual machine to create additional virtual NICs and grant access to a restricted Guest LAN, the virtual machine can couple many virtual NICs to the same Guest LAN. This scenario can lead to a DoS attack; each connection counts against the maximum allowed for the Guest LAN (and can therefore prevent others from connecting to the Guest LAN).

The VSWITCH in z/VM is a low-cost and high-performance implementation of much of the function that would be provided by a Linux virtual machine as virtual router. However, VSWITCH lacks the more advanced functions of a virtual router (such as, packet filters, bandwidth shaping, network address translation). These advanced functions can play a role in protection against intrusion and denial of service attacks. Some of this can be handled through the IEEE VLAN support in VSWITCH using an outboard VLAN-aware switch, but when additional functions are needed, you should implement a virtual router instead.

Additional z/VM resources

In this section we discuss a few other z/VM resources that require some attention to prevent a virtual machine from consuming an excessive amount of resources and impact other virtual machines.

Shared segments

Recent kernel patches (for example, the DCSS Block Device and the Execute-in-Place file system) allow Linux virtual machines to exploit z/VM Shared Segments. If we want to allow Linux to use this function, the virtual machine must be authorized to use the Diagnose 64 instruction (see “Diagnose codes” on page 7).

CMS and applications on CMS use shared segments to share code among virtual machines. Normally all virtual machines are allowed to attach the shared segment in read mode, so VM natively does not offer granular access control for shared segments. Some special purpose shared segments are defined as restricted shared segments because the data contained in these shared segments is not meant to be seen by general virtual machines. Restricted shared segments can only be attached by virtual machines with a corresponding NAMESAVE statement in their CP Directory entry.

Restriction: You might be tempted to make shared segments all restricted and use the NAMESAVE statements to control access to the shared segments (such as, to allow only some Linux servers to use some charged products). This is however not the proper way to implement access control because the NAMESAVE option also allows the virtual machine to attach the segment in non-shared mode.

If per-segment access control is required, you should run VM with an External Security Manager. For RACF/VM you would activate the VMSEGMT class and define the proper profiles and access lists.

Linux virtual machines can make use of two different type of shared segments:

[8] By default, any class G user can issue the QUERY LAN command, which may be a bit too much in an environment with many class G users.

SR The code or data in the shared segment is shared in read-only mode among the virtual machines. With the xip2 filesystem it is possible to have Linux binaries and shared libraries in the shared segment mapped directly into the Linux process address space. These programs and libraries thus need to be present in real memory only once rather than a separate copy for each virtual machine.

EW Each virtual machine that attaches an EW (Exclusive Write) segment will get a copy of the shared segment in write mode. The Linux dcss block device driver can use this segment as a writable block device in memory. This is an attractive solution for the Linux swap device.

Even when a shared segment does not contain confidential data, a Linux virtual machine can cause some performance problems by unauthorized access to shared segments:

- Segments defined with an EW part allow the virtual machine to acquire additional memory beyond the defined virtual machine size, and thus use more resources than planned.

  The EW shared segments distributed with CMS and other licensed products are small enough that there is little concern for abuse. In addition, the address range for these segments is well below the typical Linux virtual machine size (attaching these segments actually reduces resource usage).

  For the relatively large EW segments used for Linux swap space, it is wise to define them as restricted and use NAMESAVE statements in the directory for those Linux virtual machines that are allowed to use the segment.
- When a new copy of a shared segment is saved, the previous version of that shared segment (with the same name) is kept “pending purge” as long as one or more virtual machines have attached the segment. Because the shared segments for Linux tend to be rather large, keeping an excessive number of segments pending purge may fill up spool space. It is recommended to check spool utilization before saving a new segment.

System spool

Although some Linux device drivers exist to process spool files, their use is not very wide spread. Access to spool files is properly managed by z/VM (though this would require capability to issue some related CP commands). A Linux virtual machine creating spool files could cause a denial of service by filling system spool space.

Because spool files are used as a communication tool by various systems management applications, a (poorly written) application could be affected by faked spool files from a compromised virtual machine (most likely limited to a DoS).

The easy solution for this is not to provide the virtual machine with virtual spooling devices (PRT and PUN) in the CP directory. Because the **DEFINE** command is taken away as well, they can not be created either. The exception to this is the virtual console. It might be attractive to have a permanent record of the console output by spooling the device, but this requires a CP **SPOOL** command. Allowing the virtual machine to use this would also allow it to disable spooling or direct it to somewhere else (and impact other services again).

Note: It is possible to enable spooling of the virtual console in a reliable manner, using the mechanism outlined in “An alternative approach” on page 13.

Secondary Console Image Facility

SCIF allows a virtual machine to capture the console output of another virtual machine in real-time. SCIF is enabled either by a **SET SECUSER** command or by an option on the

CONSOLE statement in the CP Directory. The captured console output is frequently processed by the Programmable Operator (PROP) or Host Management Facilities (HMF) to monitor the virtual machine and optionally respond to certain messages.

The output displayed on the virtual machine console (and passed to the secondary console) in the case of Linux is all that is written to /dev/console. Normally the syslog daemon will filter the Linux system logging and only present the important data on the console. The root account on Linux could alter the syslog configuration to:

- Prevent important messages from showing on the console so that the monitoring system would fail to recognize alert messages
- Fake messages on the console that incorrectly trigger action routines
- Flood the console with a much higher message rate than usual. This could saturate the monitoring virtual machine and prevent it from responding to messages from other servers.

CP directory entry

When Linux virtual machines are unable to issue CP commands that alter the virtual machine configuration, the CP directory is the only place that defines the devices for the virtual machine. All devices that Linux needs must be defined in the directory, which makes it easy to get an overview of the resources the virtual machine can access. Devices that are not needed by Linux (and could be abused) should not be defined for the user.

Tip: The parameters on the **LOGON** or **XAUTOLOG** command can be used to override settings in the CP directory (in particular the IPL statement and the virtual machine size). This is attractive to IPL the virtual machine from a rescue kernel or use a larger virtual machine size than the default defined in the directory (but still limited by the maximum defined in the directory). This works even when the virtual machine privilege class does not allow it to issue the **DEFINE STORAGE** command.

The OPTIONS statement in the directory is used to enable various options for the virtual machine. Most of these do not apply to Linux virtual machines and thus should be avoided in the directory entry for Linux virtual machines. The following options deserve some special attention:

APPLmon This option is required when the applmon device driver is used and the virtual machine is authorized to issue Diag DC instructions.

DIAG98 This option is used by VM TCP/IP to run channel programs against network devices that bypass the translation process that CP normally does. This way TCP/IP specifies real storage addresses rather than virtual ones. A virtual machine with the DIAG98 option could basically read and write any storage in the system, which is not something you want for a guest that could be compromised. Also, Linux does not currently take advantage of this option.

Maxconn Set this value as low as you can to restrict Linux virtual machines. The default of 64 is higher than you need for most Linux virtual machines. When the Linux server is not using IUCV for its network connection, you can probably set this to 0.

QUICKDSP Use this option with care for Linux virtual machines where you know the server is not using excessive amount of CPU time and the workload justifies the virtual machine to be scheduled ahead of other virtual machines (see

Linux on IBM @server zSeries and S/390: Performance Measurement and Tuning, SG24-6926).

TODENable Use this option when the virtual machine needs to be able to set its offset on top of the system TOD clock through the **SET VTOD** command or the SCK instruction.

An alternative approach

If taking away all CP commands is too restrictive for your installation, you might consider a two-phase approach. With this approach the Linux virtual machine is initially started in class G, and is switched to a limited class L just before Linux is started in the virtual machine.

To do this you define both class G and L in the directory, which causes the guest to start up with sufficient privileges, and have it IPL CMS at startup. In the PROFILE EXEC you can do whatever is needed to prepare the userid, and conclude with setting the privilege class to L and IPL of Linux. Clearly that privilege class L should not include the **SET PRIVCLASS**, otherwise it would allow Linux to switch back to G (which would be permitted according to the directory). Because the **SET PRIVCLASS** must be done before the **IPL** command, this approach requires that class L virtual machines can issue the **IPL** command.

Note: The **SET PRIVCLASS** command is by default disabled through the *features* setting in the CP system configuration file. Some planning is recommended if you intend to use this since it takes an IPL of z/VM to activate the option.

Running CMS in the userid initially is relatively harmless because at that time the server does not yet have a network connection. The disadvantage of this approach is that it only affects the access that is controlled by privilege class. Any other resources needed for the CMS startup (such as, SFS access, shared segments) would continue to be available during Linux operation. Instead of using CMS, you could use a special "IPLer" that simply issues the necessary CP commands and passes control to the Linux kernel.

Sample MODIFY statements for the default z/VM 4.4 installation

The following **MODIFY** statements create a new privilege class L that can be used for Linux virtual machines instead of class G. This sample only applies to an unmodified z/VM 4.4 installation. The privileges granted for class L are sufficient to run a Linux 2.4 kernel with the patches from developer Works as available in March 2004. You may want to store these statements in a separate file on your CP PARM disk and imbed it in your main system configuration file:

```
Modify cmd TERMINAL      ibm G priv GL
Modify cmd QUERY subcmd TERMINAL    ibm G priv GL
Modify cmd IPL           ibm G priv GL      /* Only if reboot required  */
Modify cmd DISCONNECT    ibm O priv G
Modify cmd MESSAGE       ibm O priv G
Modify cmd LOGOFF        ibm O priv G
Modify diag nn priv G    /* Repeat for all diag codes not used by Linux */
```

The team that wrote this Redpaper

This Redpaper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Rob van der Heij is a z/VM and Linux Systems Programmer with IBM in The Netherlands. He has 20 years of experience with VM, and has worked with Linux on zSeries since the first day. His professional interest includes most software that runs on z/VM, in particular Linux. He is a regular speaker at IBM Technical Conferences and participates in Linux and VM mailing lists.

Thanks to the following people for their contributions to this project:

Gregory Geiselhart, Roy Costa, Dave Bennin, Bob Haimowitz
International Technical Support Organization, Poughkeepsie Center

Alan Altmark
IBM Endicott

Vic Cross
IBM Australia

John P Hartmann
IBM Denmark

Perry Ruiter
IBM Canada

Holger Woller, Klaus Dongus
IBM Germany

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

This document created or updated on June 9, 2004.

Send us your comments in one of the following ways:

- Use the online **Contact us** review redbook form found at:
  ibm.com/redbooks
- Send your comments in an Internet note to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYJ Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

@server®	zSeries®eveloperWorks®	Perform™
@server®	ibm.com®	RACF®
ibm.com®	z/VM®	S/390®
IBM®	zSeries®	VTAM®
Redbooks (logo) ™	C/VM™	
z/VM®	IBM®	

The following terms are trademarks of other companies:

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.