Save 10% on Exam Voucher

See Inside

# EXAM CRAM

## CompTIA®

# Cloud+

## CV0-003

Cram Sheet

Flash Cards

Practice Tests

WILLIAM "BO" ROTHWELL

EXAM CRAM

# CompTIA® Cloud+ CV0-003 Exam Cram

**William "Bo" Rothwell**

Pearson

So, to change the permissions of a file to **rwxr-xr--,** you would execute the following command:

```
chmod 754 filename
```

With octal permissions, you should always provide three numbers, which will change all the permissions. But, what if you want to change only a single permission of the set? For that, you use the symbolic method by passing three values to the **chmod** command, as shown in Table 7.1.

TABLE 7.1 **Symbolic Permission Values**

| Who | What | Permission |
|---|---|---|
| u = user owner | + | **r** |
| g = group owner | - | **w** |
| o = other | = | **x** |
| a= all sets | | |

The following example demonstrates how to add execute permission to all three sets (user owner, group owner, and others) using the symbolic method:

```
[student@localhost ~]$ ls -l display.sh
-rw-rw-r--. 1 student student 291 Apr 30 20:09 display.sh
[student@localhost ~]$ chmod a+x display.sh
[student@localhost ~]$ ls -l display.sh
-rwxrwxr-x. 1 student student 291 Apr 30 20:09 display.sh
```

# Antivirus/Antimalware/Endpoint Detection and Response (EDR)

Just about any compute system has some sort of antivirus/antimalware software available, including traditional operating systems like Microsoft Windows and Linux, as well as mobile device operating systems like iOS (for the mobile OS by Apple) and Android. They have become commonplace and are especially important on cloud resources that are often easily accessible via the Internet.

In addition to installing these products on individual systems, many organizations make use of EDR software. This software monitors and records information on the endpoints (individual systems) into a central database. This information is used to determine what risks to the network are present.

By analyzing this information, organizations can develop a better understanding of the threats that are reaching the endpoints of the organization's network.

# Host-Based IDS (HIDS)/Host-Based IPS (HIPS)

An intrusion detection system (IDS) is a software or hardware system that is designed to determine if an intrusion is occurring or has occurred on a network or a host. There are two major categories of IDS software: network-based and host-based.

A network-based IDS (NIDS) is software that monitors network packets to determine if an intrusion is taking or has taken place. A host-based IDS (HIDS) is installed on specific systems and monitors the state of the system itself to determine if an intrusion is in progress or has taken place.

There are many options available for these software programs, but the general concept is that an HIDS will utilize a database that describes what to monitor on the system. This can include monitoring actions that other software programs take, actions that users take, changes to the configuration of the operating system, and changes that are made to filesystems.

It is important to understand that an IDS is designed to monitor the system or the network for intrusions, but it is not designed to protect the system from intrusions. An IDS may take some actions, such as generating reports or sending alerts via email or SMS, but it doesn't take any direct action to protect the system or the network. It can generate alerts so an action can be taken manually.

An intrusion protection system (IPS) both monitors for intrusions and can take action if an intrusion is detected. For example, an HIPS may detect a suspicious login and, as a result, block access to the system from the source IP address.

The advantage of an IPS over an IDS is that potential threats may be neutralized quicker than if a human needed to get involved. The disadvantage of an IPS over an IDS is that false positives may result in disabling access for someone who should be allowed the access.

# Hardened Baselines

Each resource should be designed to provide specific features and functions. For example, you might have a virtual machine that is designed to serve as a

web host. Unfortunately, many operating systems have other features enabled when you perform an "out of the box" installation.

A *hardened baseline* is a set of requirements for each system you deploy. This baseline can include establishing rules on how different components of the system are configured, including the following:

- Operating system configurations
- Network configurations and services permitted on the network
- System monitoring configurations
- Data encryption methods
- Application configurations
- Security appliance configurations
- Backup policies and procedures
- Patching and update policies

The goal of hardened baselines is to establish specific rules and procedures to best ensure the security of any resource that your organization deploys. These baselines should be well established long before your organization deploys any resource in a production environment.

## Single Function

As mentioned in the previous section, each resource should be designed to perform a single function. For example, a virtual machine that provides an organization's web server should not also function as the organization's mail server, NTP server, and file server.

The separating of functions is one of the primary advantages of cloud computing and a major reason to consider utilizing containers rather than full operating systems. In a cloud infrastructure you can customize the system resources (CPU, memory, and so on) to meet the need of the compute resource that you are deploying. You don't need to guess how much system resources your web server will need, and you don't have to feel compelled to "fill up" an expensive server with additional services.

By using a single function methodology, you provide a more secure environment. For example, maybe an attacker can hack into your web server (granted, this is bad), but if you use a single function methodology, the attacker doesn't automatically gain access to your mail server, NTP server, and file server.

# File Integrity

See "Integrity" in Chapter 8, "Data Security and Compliance Controls in Cloud Environments."

# Log and Event Monitoring

See "Monitoring" in Chapter 16, "Logging, Monitoring, and Alerting."

# Configuration Management

See the following sections:

- "Hardening and Configuration Changes" in Chapter 6, "Secure a Network in a Cloud Environment."
- "Configuration Management Database (CMDB)" in Chapter 17, "Operation of a Cloud Environment."

# Builds

When an organization develops software and the software is released, either internally or externally, the release is referred to as a *build*. This section explores different types of builds.

## Stable

A *stable build* is designed to be a release that is ready for a production environment. Typically, if you are a customer who purchases software, that software is considered a stable build.

Prior to a software build being released, earlier releases are called *beta builds*. Some organizations like to have access to beta builds because this access provides them with early insight as to how the software will perform. However, beta builds come with little to no support or warranty. They are considered "use at your own risk" and should not be used in production environments.

## Long-Term Support (LTS)

A *long-term support build* is a stable build that should be supported for a longer than average period of time. This can be an important issue for some organizations because moving to a new version of software can pose several challenges, including:

- Time, effort, and money to ensure that the new version performs within standards.
- Potential new licensing costs.
- The need to deal with potential inconsistencies or incompatibilities. For example, a newer version of software might not integrate with other software that the organization is already using.
- Additional training costs to teach existing employees and customers how the new version of the software behaves.
- Reduced production as employees attempt to use the new version of the software.

One disadvantage of utilizing an LTS build is that new features that are released with the regular stable build are not normally implemented in the LTS build.

## Beta

See the earlier "Stable" section.

## Canary

You may have heard how miners would take a canary bird into the mines with them to determine if the air held dangerous levels of toxic gases. The idea was that the canary had a faster breathing rate than humans and would show signs of the presence of toxic gases quicker than humans would.

A *canary build* works on a similar concept. New features are released to a specific set of beta testers (pilot users) to determine if the new features have any negative impact on the software. The features are provided in the new beta builds in a very specific manner and typically spread out over several beta releases. This type of build allows the developers some insight as to which new features may have caused an issue and allows the developers the time to fix the issues before releasing the software in a stable build.

# Operating System (OS) Upgrades

See “Scheduled Updates” in Chapter 9, “Security Requirements.”

# Encryption

See “Encryption” in Chapter 8.

## Application Programming Interface (API) Endpoint

An API is a technique that is used to provide a well-known communication method between a client and a server. A client will issue an API request to a server, typically using representational state transfer (REST; see “Authorization” in Chapter 23, “Troubleshoot Security Issues”).

API calls across the Internet are common. Even if the API call is made within an enclosed network, it is important that the transmission itself be encrypted. For this reason, the endpoint of API connections is normally established with the HTTPS protocol.

## Application

Applications often store and transmit data. To ensure the security of this data, all data in transit and all data at rest should be in an encrypted format. This is especially important in cloud environments because data is often more accessible as it is stored in a network-accessible location.

## OS

Typically, in regard to OS encryption, what is really being encrypted is the storage disk or the filesystem where the OS resides. See the “Filesystem” section later in this chapter.

## Storage

Most cloud-based storage devices provide some level of encryption although it might not be enabled by default. Cloud storage that is used to hold sensitive data should have the encryption option turned on.