COMPSCI 750 – Assignment 5
Gunnar Yonker

1.

(1) Secure Operating System: Focuses primarily on security features. These systems are often designed from the ground up with security principles in mind, such as the principle of least privilege, mandatory access controls, and strong separation between user space and kernel space. Examples include SELinux, Trusted Solaris, and Multics.

Commercial Operating System (Like Linux or Windows): These are designed for general use by consumers and businesses. While they have security features, their primary design goal is often usability, compatibility, or performance. Over the years, security has become a bigger concern, leading to improvements, but it might not be the central focus.

Many commercial operating systems have heavily invested in integrating security features and technologies which has led to a narrowed gap between purely secure OS and commercial OS over time.

(2) By associating labels with both data and processes, a system can enforce rules based on these labels. For instance, data classified as "confidential" might only be accessed by processes with an equivalent or higher-level security label. The Biba Integrity Model would be appropriate. The Biba model prevents information flow from lower-integrity subjects to higher-integrity subjects (no read up, no write down). This ensures that a high-integrity process (good code) doesn't become tainted by lower-integrity data (bad code or data).

(3) A reference monitor is an abstract concept in security, representing a mechanism that mediates all access to objects by subjects, ensuring that the enforcement of security policy is uncompromised. To ensure that trustworthiness of a reference a few steps must be followed:

It must be tamper-proof, meaning it cannot be modified by unauthorized entities.

It must always be invoked and be the only way to access protected objects.

It should be small enough to be subject to thorough testing and verification, ensuring that it's free of flaws.

(4) When a high privilege process accesses a low privilege file, care should be taken to ensure that the high privilege process doesn't elevate the privilege of the low privilege data inadvertently, or that it doesn't get tricked by malicious low-privilege data. The Bell-LaPadula Model handles this behavior particularly well. It is primarily concerned with data confidentiality and employs mandatory access controls. In this model, there is a "no read down, no write up" (or "ss-property" and "ds-property") policy. This ensures that while a high privilege process can read a low privilege file, it cannot write to it, preventing the potential for elevating the privileges of the data in the file.

COMPSCI 750 – Assignment 5
Gunnar Yonker

2.

(1)

p is denied for accessing:

Any ring outside the bracket will be denied access. Hence, p should be executing in rings: 0, 1, 2, 6, 7.

p is allowed to read, write, or execute:

Within the access bracket, all operations permitted by the ACL can be performed. As p has full access rights, it can read, write, and execute in rings: 3, 4, 5.

p is allowed to read or execute but not write:

Since the ACL gives p full rights, there isn't a ring in the bracket where p can't write. However, in a general sense and without considering the ACL, rings that allow reading or executing without writing would be at the boundaries of the bracket. Here, the boundaries are 3 and 5. But given the ACL, there's no restriction on writing.

(2) Biba's strict integrity policy model employs the idea of "no read down" and "no write up". This means:

A subject at a high integrity level cannot read an object at a lower integrity level ("no read down").

A subject at a low integrity level cannot write to an object at a higher integrity level ("no write up").

Multics' Protection Ring:

A process in a higher ring (like ring 2) can access a segment with an access bracket of (3,5) for reading or executing (reading down), violating Biba's "no read down".

A process in a lower ring (like ring 6) can't write to a segment in a higher access bracket (like 3,5), which is consistent with Biba's "no write up".

A process in ring 3 accessing a data segment with an access bracket of (2,4) can read and execute but can't write to it. This violates Biba's "no write up" since a lower ring is prevented from writing to a higher bracket.

In conclusion, while the Multics ring protection mechanism provides a graded level of security and access control, it does not strictly adhere to the Biba integrity model. The concept of rings and brackets is more flexible but also more complex, making it harder to ensure a mandatory protection state. This is evident from the lecture slides, indicating Multics' failure to meet mandatory protection state and reference monitor guarantees. Still, that does not mean Multics wasn't secure; it just took a different

approach than what the strict Biba model proposes. Multics provides more flexibility, but this can sometimes lead to scenarios where integrity can be compromised, unlike the Biba model.

3.

(1)

p1 cannot invoke:

If p1 is outside of p2s's call bracket, then p1 cannot invoke p2. In this case, p1 should be in rings 0-3 or 7.

p1 can invoke p2 but a ring-crossing fault occurs:

This means that p1 is trying to invoke p2 in a ring which is less privileged than p2's access bracket. For this case, p1 should be in ring 5.

p1 can invoke p2 if a valid gate is used as an entry point, a ring-crossing fault occurs:

This means that p1 is trying to invoke p2 in a more privileged ring than p2's access bracket. In this case, p1 should be in ring 1.

P1 can invoke p2 and no ring-crossing fault occurs, no gate is needed:

This means that p1 is invoking p2 within p2's access bracket. For this case, p1 should be in rings 2-4.

(2) In case b, process p1's ring number will increase (go to a less privileged ring) when a ring-crossing fault occurs. In case c, process p1's ring number will decrease (go to a more privileged ring) when a gate is used for entry.

Biba's low water mark policy focuses on ensuring the integrity of information. If a subject reads data at a lower integrity level, then the subject's integrity level is lowered to the integrity level of that data.

Multics policy is not quite the same as the low water mark in Biba. The ring-crossing in Multics focuses on the privilege of a process and how it can interact with other processes based on hierarchical protection rings. The low water mark policy in Biba focuses on the integrity level of subjects and objects, adjusting a subject's integrity level based on the data it interacts with. While both mechanisms deal with changing privilege or integrity levels, they're based on different principles and objectives. Multics' rings are about controlled access and compartmentalization in hierarchical levels, while Biba's integrity levels are about preserving the integrity of information.

COMPSCI 750 – Assignment 5
Gunnar Yonker

4.

(1) Four Major Components in Android Application Framework

Activity: This is the user interface component, and each activity is considered as a “screen”. Activities can start other activities either within the same application or in another application.

Service: Responsible for background processing like playing music or fetching data. Services can be local or remote and allow for communication between applications.

Content Provider: It provides a standardized interface for sharing data between applications. It models the data similarly to a relational database, allowing for operations equivalent to SELECT, UPDATE, INSERT, and DELETE.

Broadcast Receiver: These components act as handlers for events or messages. They “subscribe” to specific actions and are automatically called by the system when those actions occur.

(2) Kernel security in Android is enforced through the Linux Security model. Each application in Android runs as a unique user with a unique UID, enabling process isolation. This ensures that users can’t read other users’ files. A user cannot exhaust another user’s memory, CPU resources, or devices.

(3) At the application level, security is enforced through the Android permission model. Permissions are set at install time and are immutable unless the application is reinstalled. Each application requests a list of permissions, and the Android middleware contains a reference monitor that checks these permissions during inter-component communication (ICC). Only if the required permissions are granted, the ICC is established.

(4) Mandatory Access Control (MAC) in Android is primarily implemented through the assignment of permission labels to applications and components. When a component initiates the ICC, the reference monitor checks the permission labels of the initiating application. If the target component’s access permission label matches the initiating application’s labels, the ICC is allowed. This model is mandatory since all permission labels are set at install time and remain unchanged unless the app is reinstalled.

(5) The manifest file, AndroidManifest.xml, is a file that describes the contents of an Android application package. It lists the components within the application, specifies access rules, and defines runtime dependencies. In terms of application security, it specifies the permissions an application requires to run. It lists the components, ensuring unauthorized components cannot execute. Lastly, it defines access control policies for component access.

COMPSCI 750 – Assignment 5
Gunnar Yonker

(6) Security Refinements in Android:

Private Components: Reduces attack surface by defining components are private.

Benefit: Enhances security by limiting component access.

Downside: May limit inter-app functionalities if not used judiciously.

Implicitly Open Components: If a public component doesn't have an explicit permission, any app can access it.

Benefit: Simplifies development when access permissions are unclear.

Downside: Can introduce security risks if overlooked.

Broadcast Intent Permissions: Protects the broadcasting of intents.

Benefit: Enhances privacy.

Downside: Adds another layer of complexity for developers.

Content Provider Permissions: Defines read or write permissions for content.

Benefit: Provides granular control over content access.

Downside: Might confuse developers if not documented properly.

Protected APIs: Direct API access is protected with permission checks.

Benefit: Protects sensitive system resources.

Downside: Adds another layer of permissions developers must be aware of.

Permission Protection Levels: Control how developers assign permission labels.

Benefit: Ensures sensitive functionalities are accessed only by specific apps or the framework.

Downside: Can be restrictive for third-party app developers.

Pending Intents: Allows better integration between system and third-party apps.

Benefit: Enhances inter-app communication and integration.

Downside: If misused, can introduce security vulnerabilities.