DSCI 525 - Trusted System Design, Analysis, and Development Project

Rishit Saiya (rsaiya@usc.edu)

I. Literature of TCSEC

The Trusted Computer System Evaluation Criteria (Orange Book) served as a prime standard for assessing the security of computer systems within the US Department of Defense. It introduced a framework of security criteria to evaluate the security posture of these systems, categorizing them into different assurance levels ranging from D to A, with A1 representing the highest level of security. The TCSEC emphasized the importance of a secure operating system, advocating for the use of a security kernel.

In order to evaluate the security of a system, TCSEC outlined 25 factors, and a system's rating was determined by how many of these factors it satisfied. Therefore, an A1 system would meet all 25 factors. The TCSEC framework focused on secure operating systems and laid the groundwork for the Trusted Network Interpretation (TNI), which extended these principles to computer networks. The factors included requirements such as security policy and design documentation, highlighting the comprehensive approach taken to ensure the security of computer systems.

Regarding security policy and formal design documents, I will briefly outline the expectations for an A1 system's security policy. An A1 system must implement Mandatory Access Control (Discretionary Access Control is optional) for both confidentiality and integrity. It is essential to have a FSPM that includes a mathematical proof supporting the security policy and demonstrating a 1:1 mapping between security policies and mechanisms. Additionally, a FTLS encompassing functions' definitions performed by the TCB level and with both HW/SW mechanisms supporting the policies is needed. A formal analysis technique should be incorporated to detect any covert channels.

1. Object Reuse: Before reallocating storage objects to the TCB's unused pool, all authorizations to the object's information must be revoked. Once an object is released, no information from previous actions should be accessible to new subjects. The TCB ensures that its controlled storage objects do not contain unauthorized information before granting access.
2. Labels: All system resources (subjects and objects) have associated sensitivity labels for enforcing Mandatory Access Control (MAC) for confidentiality and integrity. When using external non-labeled data, the TCB requests and receives the sensitivity level from an authorized user. If no label is provided, the system assigns a label to the imported unlabeled data based on the device labels used for importing.
3. Label Integrity: The labels of sensitivity should precisely reflect the confidentiality and integrity levels of different types of subjects and objects. This concept relates to label stability,

where object sensitivity labels should remain constant. When exporting an object, the associated label should also be exported.
4. Exportation of labeled objects: The TCB classifies each communication channel and input/output device as either single-level or multilevel, with any manual alterations being subject to audit.
5. Exportation to Multilevel Devices: This factor pertains to scenarios involving cross-domain solutions where system objects are shared with multilevel devices trusted to manage objects of varying security levels. Sensitivity labels associated with objects must be shared to enforce Mandatory Access Control (MAC). When exporting an object to a multilevel input/output device, its label of sensitivity must also be exported, maintained on the similar physical media as the exfiltrated data, and in the same format. The protocol utilized on the channel must allow for unanimous pairing between labels of sensitivity and different types of objects.
6. Exportation to Single-level Devices: Unlike the previous factor, this one addresses single-level communication channels that cannot be relied upon to segregate objects with different sensitivity levels. As a result, it is not required to maintain sensitivity labels for data processed by single-level I/O devices and communication channels. Nevertheless, the TCB needs a dependable means of communicating with an authorized user to indicate the single sensitivity level of data imported or exported through these devices.
7. Labeling Human-Readable Output: System administrators must specify the printable label names of objects. By default, all human-readable output, including hardcopy items like pages and softcopy items like documents and graphics, must have human-readable sensitivity labels.
8. Subject Sensitivity Labels: If the sensitivity level linked to a terminal user changes while using a session, the TCB must notify the user. Upon request, the complete sensitivity label for the subject must be displayed by the TCB.
9. Device Labels: The TCB needs to designate both minimum and maximum sensitivity levels for all connected physical devices. These levels are crucial for implementing MAC constraints and act as a way to limit the sensitivity levels of information allowed in the physical environment where the devices are situated. For instance, sharing an object with a specific sensitivity level would not be possible with a device whose range lacks a dominant level.
10. Mandatory Access Control (MAC): TCB must enforce Mandatory Access Controls, governing access control based on hierarchical sensitivity levels (such as unclassified, secret, etc.) and non-hierarchical categories (usually representing the field or domain of the resource) for all subjects and objects. Access to objects is determined by using simple security and *-security properties. The system should have more than two access classes and enforce MAC for confidentiality and integrity. Additionally, the system should have an authentication mechanism to accurately identify users and their corresponding access class.
11. Trusted Path: The trusted path is a channel of secure communication between users and TCB, ensuring that no malware (e.g., Trojan horse, virus) is present in the system during user interactions. Typical use cases of the trusted path include user login and changing access classes

of system resources. The TCB must explicitly inform users when the trusted path is active, for example, by using a special light or a different terminal.

12. System Architecture: This aspect focuses on the design and computational structure of the TCB. To enhance resistance to tampering, the TCB should operate in a separate execution domain. It should enforce process isolation by allocating each process its address space. The TCB should be hierarchically organized into independent modules with layering, abstraction, and data hiding principles, akin to a loop-free hierarchical structure with each layer represented as Parnas modules. Leveraging hardware support, such as segmentation, can enforce access control on objects. Additionally, the TCB should adhere to the principle of minimization by including only security-relevant and critical modules in the TCB, with other modules implemented outside the TCB. The principle of least privilege should also be implemented by the TCB, and its user interface should be well-defined.

13. System Integrity: HW/SW capabilities should be included to verify the proper computations of on-site hardware/firmware of the TCB periodically. This typically involves running diagnostics, checking boot time, and keeping software up-to-date.

14. Covert Channel Analysis: Formal analysis methods should be employed to examine the system for the presence of covert channels and calculate the maximum bandwidth of any identified channels.

15. Trusted Facility Management: The TCB should have a separate operation and sysadmin roles in order to facilitate their distinct operations from generic types of users. These roles should have distinct access labels, and their responsibilities should be clearly outlined. Their actions should be auditable under the security admin role and should be limited to those which are necessary for carrying out security-relevant functions.

16. Trusted Recovery: The TCB should support a mechanism for securely restoring the system to its last working state in the event of a crash or shutdown without compromising security.

17. Security Testing: The TCB should undergo security testing to identify vulnerabilities, flaws, and any system state that violates security policies. It should be tamper-proof and resistant to subversion, demonstrating consistency with mapping FTLS to TCB's source code to ensure a 1:1 mapping between security policy and implemented mechanisms.

18. Design Specification and Verification: FTLS specification documents are to be prepared in order to outline that particular system's security policy in place, in order to demonstrate the 1:1 mapping between security policy and implemented mechanisms, and mathematically prove that the policy model does not lead to security vulnerabilities. The FTLS should be consistent with the TCB.

19. Configuration Management: A configuration management system should track all design documents (e.g., formal top-level spec, descriptive top-level spec, testing document) throughout the TCB's Software Development Life Cycle (SDLC). It should also track the source and machine code of the TCB, preserving the confidentiality and integrity of these documents and software.

20. Trusted Distribution: To ensure the integrity of the system distributed to end-users, a trusted distribution facility must be used. This feature guarantees that the mapping between the "master data describing the current version of the TCB and the on-site master copy of the code for the current version" remains unchanged during delivery to the end-user.
21. Security Features User's Guide: A document should be created to explain the security mechanisms implemented by the TCB, detailing the threats they mitigate and how they are used.
22. Trusted Facility Manual: This manual is intended for system administrators, providing guidance on operating the system, controlling implemented security controls and mechanisms, understanding auditing processes, and conducting day-to-day system administration tasks. It should also include information on generating the TCB from the source code.
23. Test Documentation: This document outlines the test plan, test functions and methodology, and their results. It should cover testing of all security mechanisms, with the report audited by evaluators. Covert channel analysis should be part of the assessment, along with mapping between the FTLS and the TCB's source code.
24. Design Documentation: This document presents the security model, demonstrating how it is secure and addresses all security vulnerabilities. It explains how the TCB follows RVM and adheres to its principles (tamperproof, always invoked, and verifiable). It should also detail how the TCB satisfies the security model, including explanations of internal mechanisms not covered in the FTLS.
25. Rating Maintenance Phase (RAMP): This factor involves arranging appropriate tools, mechanisms, procedures, and personnel to comply with updates in the rating program of Trusted Evaluation Programs.

These were the 25 criteria established by TCSEC/TNI for evaluating a system, particularly focusing on security kernels. However, a limitation of security kernels is that certain security-relevant functions, such as updating security databases and system backups, cannot be implemented within the kernel itself due to the principle of minimizing kernel functionality. Implementing these functions outside the kernel avoids the need for rigorous testing and verification. An alternative approach is the separation kernel, which emulates a distributed environment with multiple isolated partitions running on top of it. All communication between partitions must pass through the separation kernel. This allows security-relevant functions to be implemented within a partition as a trusted service without increasing the complexity of the separation kernel. Nevertheless, these trusted services still require evaluation.

II. Literature of Separation Kernel Protection Profile (SKPP)

The Separation Kernel Protection Profile (SKPP), influenced by the Common Criteria, was developed as a protection profile for evaluating systems based on separation kernels. When referencing specific details from a document [3], paragraph numbers cited in that document will be used (preceded by [3]). According to SKPP, a separation kernel partitions the system into divisions, with each partition typically representing a security policy-equivalence class. The separation kernel controls inter-partition flow, and authorized information flow is established during system boot-up [3][7]. SKPP specifies various requirements for separation kernels, such as enforcing information flow control, isolating resources, ensuring trusted initialization, delivery, and recovery, and implementing audit mechanisms [3][8]. The core functional requirements of a separation kernel, as outlined in SKPP [3][37], are: The Separation Kernel Protection Profile (SKPP) outlines several key requirements for separation kernels in systems. These requirements are safeguarding resources, such as CPU/memory and devices from untrusted access. This ensures that internal resources remain isolated and protected. Furthermore, the SKPP requires the partitioning and isolation of exported resources, ensuring that each partition operates within its designated boundaries. The separation kernel should also control information flows between partitions and exported resources, permitting only authorized flows. Finally, the SKPP mandates the provision of audit services, enabling the system to monitor and log security-relevant events for later analysis. The aforementioned functional requirements loosely translate to security mechanisms [3][42] which are: The Separation Kernel Protection Profile (SKPP) also specifies additional requirements for separation kernels in systems. These requirements include implementing information flow control to ensure strict partition isolation. This ensures that the TSF is not compromised during its operation. Additionally, separation kernels must provide trusted initialization and recovery functions to ensure the system can be securely started up and recovered from failures. They should also be able to detect and respond to security function failures to maintain the system's security posture. Finally, separation kernels must generate audit data to enable the monitoring and analysis of security-relevant instructions within the given system.

SKPP defines two main categories of security requirements which are Functional and Security Assurance requirements. The former comprises essential requirements that the separation kernel must meet, while the latter consists of additional security measures that the kernel may adopt to enhance robustness and assurance. The security functional requirements encompass six categories: Security Audit, User Data Protection, Identification and Authentication, Security Management, Protection of the TSF, and Resource Utilization. On the other hand, the security assurance requirements include eight categories: Configuration Management, Delivery and Operation, Development, Guidance Documents, Life Cycle Support, Ratings Maintenance, Platform Assurance, and Testing, and Vulnerability Assessment. One of the critical security requirements in SKPP is ensuring the confidentiality and integrity of the configuration flow vector, which determines the inter-partition flow and allocation of subjects and objects to partitions. If an adversary gains access to this vector, the entire system could be compromised.

III. Comparison of TCSEC kernel evaluation requirements against SKPP

The document cross-references the TCSEC criteria with the corresponding criteria in SKPP, if applicable. All references provided below are to specific paragraph or section numbers in SKPP [3].

1. Object Reuse: SKPP addresses this aspect under "Residual Information Protection," outlined in the Security Objectives as O.RESIDUAL_INFORMATION and detailed in the Security Functional Requirements in section 5.2.3.
2. Labels: In separation kernels, partitions are associated with equivalence classes, which correspond to TCSEC labels, specifically MLS sensitivity levels [3][141].
3. Label Integrity: While not explicitly mentioned in SKPP, there is a related security objective, O.CORRECT_CONFIG, with corresponding security assurance requirements, ADV_CTD_EXP.1 and AGD_ADM_EXP.1, detailed in sections 6.3.2.1 and 6.4.1.1. respectively. This requires that separation kernels have mechanisms to ensure the accurate generation of initial configuration vectors, which include the equivalence classes or labels.
4. Exportation of Labeled Information: SKPP does not include specific evaluation criteria for exporting labeled information.
5. Exportation to Multilevel Devices: SKPP does not include specific evaluation criteria for exporting to multilevel devices.
6. Exportation to Single-Level Devices: SKPP does not include specific evaluation criteria for exporting to single-level devices.
7. Labeling Human-Readable Output: SKPP does not include specific evaluation criteria for labeling human-readable output. However, ADV_CTD_EXP.1.3D in section 6.3.2.1 discusses representing the initial configuration vector in a human-readable format. This includes details such as the system's partitions, information flow policy, and the labels of partitions and their corresponding subjects or objects. While this requirement differs somewhat from TCSEC, it enables humans to understand the relationships between partitions and their equivalence classes, as well as which subjects or objects belong to each partition.
8. Subject Sensitivity Levels: SKPP does not have a specific requirement for subject sensitivity levels. However, section 5.5.2.1 (FPT_CFG_EXP.1) mentions an optional requirement that separation kernel systems may choose to implement. This requirement allows for changes to the configuration vector of the system, including the subject's equivalence classes (sensitivity labels), while the system is running and visible to the user. However, it does not explicitly state whether a subject can query their own equivalence class at any given time.
9. Device Labels: SKPP lacks specific evaluation criteria for device labels.
10. Mandatory Access Control: SKPP does not provide criteria for implementing Mandatory Access Control (MAC) in separation kernels. Unlike TCSEC, which mandates MAC policy for A1 systems for all subjects and objects to preserve both confidentiality and integrity, SKPP does not mention MAC, the BLP/Biba model, or their simple security and *-property. Therefore, separation kernels are not required to implement MAC for access control policy according to SKPP. Instead, it is up to individual developers to decide what policy they want to implement.

SKPP mentions that a partition flow policy must be devised for the system to enforce access control, but it does not specify that it must be MAC, unlike TCSEC.

11. Trusted Path: SKPP does not include evaluation criteria for a trusted path. This is likely because all inter-partition flow must go through the separation kernel, meaning that for any object sharing or data flow, subjects would indirectly go through the separation kernel. However, if a subject wants a direct line of communication with the separation kernel for certain purposes, such as secure login or other security essential functions, it is not possible since SKPP does not provide for a trusted path.

12. System Architecture: SKPP includes several security objectives and functional requirements related to system architecture. For example, Security objective O.REFERENCE_MONITOR specifies that the separation kernel will enforce a reference validation mechanism in a separate execution domain. This requires the separation kernel to satisfy the RM principles which are tamperproof, non-bypassable, and verifiable. Security assurance requirement ADV_INT_EXP.3 in section 6.3.7.1 includes several sub-points related to the architecture of the separation kernel. For instance, it specifies that the kernel should be designed as a modularized layered system, while enforcing the principle of least privilege and minimization. However, it does not explicitly mention data hiding or abstraction.

13. System Integrity: The security objective O.CORRECT_TSF_OPERATION in SKPP aligns with the system integrity factor in TCSEC. Security requirements for this objective in SKPP include FAU_ARP.1, FMT_MOF.1, FMT_MSA_EXP.1, FMT_SMF.1, FPT_AMT.1, and FPT_TST_EXP.1.

14. Covert Channel Analysis: SKPP includes a security objective O.COVERT_CHANNEL_ANALYSIS that mandates separation kernel systems to undergo analysis for covert channels. Corresponding security requirements for this objective are FDP_IFF.3 and AVA_CCA_EXP.2. The former, as noted in 5.2.2.2, imposes a bandwidth limit on covert channels, while the latter, specified in 6.9.1.1, requires systems to undergo inter-partition covert channel analysis.

15. Trusted Facility Management: While SKPP 2.7.1 does not explicitly require support for administrative roles and security management functions, the security objective O.MANAGE mandates systems to support functions for system administrators to manage the system's configuration data and security functions. Corresponding security requirements for this objective are FMT_MOF.1, FMT_MSA_EXP.1, FMT_MSA_EXP.3, FMT_MTD.1, FMT_MTD.3, FMT_MCD_EXP.1, and FMT_SMF.1.

16. Trusted Recovery: The security objective O.RECOVERY_SECURE_STATE serves the same purpose as TCSEC's trusted recovery factor. Corresponding security requirements for this objective include FPT_FLS.1, FPT_MTN_EXP.1, FPT_MTN_EXP.2, FPT_RCV.4, and FPT_RCV_EXP.2, ensuring that the system can recover without compromising security when it goes down.

17. Security Testing: SKPP section 6.8 addresses the testing requirements for the system, encompassing an analysis of test coverage (ATE_COV.3), depth of testing (ATE_DPT.3), functional testing (ATE_FUN.2), and independent testing (ATE_IND.3).
18. Design Specification and Verification: SKPP includes several security requirements pertaining to this factor. ADV_FSP_EXP.4 (in section 6.3.3.1) mandates a formal representation of the functional specification of the separation kernel. ADV_HLD_EXP.4 (in section 6.3.4.) focuses on a semi-formal high-level design of the entire system. ADV_LLD_EXP.2 (in section 6.3.8.1) specifies requirements for a semi-formal low-level design. ADV_RCR.3 (in section 6.3.10.1) emphasizes formal correspondence, while ADV_SPM.3 (in section 6.3.11.1) needs a formal security policy model and a formal match between the functional specification and the security policy.
19. Configuration Management: The security objective O.CHANGE_MANAGEMENT addresses configuration management. Corresponding security requirements include ACM_AUT.2, ACM_CAP.5, ACM_SCP.3, ALC_DVS.2, ALC_FLR.3, and ALC_LCD.2. ACM_AUT.2 (defined in section 6.1.1) specifies that the system should have a configuration management tool that automates and tracks changes in the configuration of the entire system.
20. Trusted Distribution: SKPP uses the term "trusted delivery" instead of "trusted distribution." The security objective O.TRUSTED_DELIVERY and its corresponding requirement ADO_DEL_EXP.2 (defined in section 6.2.1.1) establish cryptographic measures for trusted delivery, such as employing message authentication codes, digital signatures, hashing and so on.
21. Security Features User’s Guide: SKPP's User Guidance security requirement (AGD_USR.1 defined in section 6.4.2.1) aligns with this TCSEC requirement.
22. Trusted Facility Manual: Security requirement AGD_ADM defined in section 6.4.1 addresses the security manual for system administrators.
23. Test Documentation: The security requirement Testing defined in section 6.8 of SKPP pertains to the documentation of testing (functional testing, security testing, independent testing, etc.).
24. Design Documentation: The security objectives O.SOUND_DESIGN and O.SOUND_IMPLEMENTATION focus on having documentation for design specifications and their alignment with the security policy model. Several security requirements relate to these objectives: ADV_ARC_EXP.1, ADV_FSP_EXP.4, ADV_HLD_EXP.4, ADV_INI_EXP.1, ADV_INT_EXP.3, ADV_LLD_EXP.2, ADV_RCR.3, ADV_SPM.3, APT_PDF_EXP.1, APT_PSP_EXP.1, AVA_SOF.1, APT_PSP_EXP.1, and ADV_IMP_EXP.3. These requirements collectively address TCSEC's criteria for design documentation.
25. RAMP: SKPP's security objective O.RATINGS_MAINTENANCE concerns changes and updates in the system's ratings.

Hence, among the 25 TCSEC kernel evaluation criteria, SKPP only fulfills 17, and even among those, some do not precisely align with TCSEC's requirements – SKPP has somewhat relaxed the strict conditions. Among the requirements that differ, two crucial ones are MAC and Trusted

Path. TCSEC clearly defines MAC and Trusted Path as essential and emphasizes their significance; however, SKPP does not mention them. While SKPP states that a security policy will enforce information flow between partitions, it does not mandate that the policy be MAC. However, SKPP includes certain additional security considerations not found in TCSEC. For example, SKPP addresses physical security threats (OE.PHYSICAL), platform threats (dependent on the platform for which the separation kernel system is implemented [3][87]), misconfiguration threats in the Inter-Partition Flow Policy, and also includes provisions for Vulnerability Assessment Threats (O. VULNERABILITY_ANALYSIS_TEST). In these scenarios, SKPP addresses external non-system threats that could arise from the environment. As the separation kernel manages inter-partition flow, it includes security requirements in order to maintain the integrity of the static config file, which determines the allocation of subjects and objects to partitions and controls inter-partition flow. This file is crucial for the separation kernel; if it is tampered with, an adversary might access data in a higher equivalence class partition from a lower class partition. SKPP also includes explicit requirements for subject identification and authentication, as well as audit systems, which were not present in TCSEC. SKPP imposes strict conditions on how the initial configuration file should be handled and modified. It specifies that static or dynamic modification of the configuration file should only be done by authorized subjects and only in maintenance mode. Maintenance mode is a specialized mode in separation kernel systems that allows the system to be modified and updated securely, reverting to the last secure state upon exiting maintenance mode. TCSEC did not have such a mode to enable secure updates or backups. SKPP includes several good security requirements for separation kernel systems. However, the absence of essential conditions like MAC and Trusted Path means that even if a system is evaluated to be 6+ according to SKPP, it will not meet the criteria for an A1 trusted system. The absence of MAC could lead to security vulnerabilities.

IV. Literature of GEMSOS

Gemini Trusted Network Processor (GTNP) is a system rated A1, evaluated against TCSEC/TNI criteria. Written in Pascal, GTNP implements a reference monitor with Mandatory Access Control (MAC) as a security kernel (RVM). It meets all 25 TCSEC/TNI criteria with high assurance, earning its A1 rating. GTNP employs various hardware and software techniques, including segmentation, hardware rings, data hiding, layering, and MAC, to provide security. The GTNP kernel incorporates several security mechanisms, such as a multilayer segment naming system, a rigorous hierarchical ring mechanism, and restricted device access. It utilizes the Data Encryption Standard (DES) for encryption. GTNP can serve as a foundation for building other applications, including general-purpose secure operating systems and network-oriented programs like network elements, guards, terminal access controllers, gateways, and packet switches.

V. GEMSOS satisfying criteria TCSEC/TNI criteria

The following is referenced from [3][section 8]:

1. Object Reuse: Gemsos addresses information leakage vulnerabilities during object reallocation. Disk storage media is overwritten before allocation to a subject, and segment contents imported from secondary storage are overwritten in main memory. Hardware registers are cleared between context switches, and segments and I/O buffers are treated similarly. The math coprocessor is also cleaned when necessary.
2. Labels: Gemsos assigns labels to all system resources (subjects and objects) to enforce Mandatory Access Control (MAC) based on the Bell-LaPadula (BLP) and Biba models.
3. Label Integrity: GTNP uses a cryptographic checksum obtained from the creating subject to ensure label accuracy for objects created on media accessible through I/O devices. The kernel's internal data structure protection ensures subject labels, and the Trusted Computing Base (TCB) ensures the label falls within the volume's range before creation. This cryptographic checksum is crucial because it's not verified when an existing item is read, as a volume could be treated as a raw disk when not mounted. Additionally, due to the single-level nature of all import/export devices in GTNP, labels for subjects and objects are never exported.
4. Exportation of Labeled Information: All devices and communication channels used for import and export in GTNP are single-level, with designated minimum and maximum ranges for each channel and device.
5. Exportation to Multilevel devices: GTNP does not include support for multilevel devices in its evaluated configuration.
6. Exportation to Single-level devices: Each device in GTNP is single-level, with a degenerate range where the minimum label equals the maximum label. This means that data exported by a single level device is by default assigned the label of sensitivity of the device.
7. Labeling Human-Readable Output: GTNP does not include any human-readable output devices.
8. Subject Sensitivity Levels: GTNP does not support interactive users during operation.
9. Device Labels: GTNP assigns minimum and maximum security labels to each physical device, limiting the levels at which logical devices can be attached based on the explicit labels of physical devices. Import/export devices specify the permitted range of topics to which they may be attached. Access to a device requires that the subject's label encompasses the range of the attached device, following Mandatory Access Control (MAC) requirements for successful attachment.
10. Mandatory Access Control: GTNP enforces a mandatory access control policy using Biba for integrity and Bell-LaPadula (BLP) for confidentiality. The Non-Discretionary Security Manager (NDSM) layer of the kernel is always invoked to determine if the subject-object labels have a dominant relationship before granting access. Labels consist of up to 96 non-hierarchical categories and 2 hierarchical levels, encompassing secrecy and integrity components.

11. Trusted Path: GTNP does not support direct user input, making the trusted path requirement irrelevant.
12. System Architecture: Gemsos implements a combination of hardware security mechanisms, software engineering methods, and design documentation to meet system architectural requirements. GTNP utilizes hardware rings and segmentation for hardware support, operating at privilege levels inaccessible to untrusted applications. The kernel is organized as a reference validation mechanism (RVM), following principles of modularization, layered architecture, data hiding, and abstraction. Process isolation is achieved through segmentation, privilege level mechanisms, and hardware task switching. GTNP multiplexes the last 3 privilege levels into 8 rings. No untrusted applications may run in PL0 (hardware privilege level 0), where the kernel runs. R2 in PL1 is the bare minimal ring/PL for an untrusted process. For kernel organization, which is a RVM, Gemsos uses the principles of modularization, layered architecture, data hiding and abstraction. Its kernel is composed of 14 layers with each layer being a Parnas module.
13. System Integrity: GTNP performs hardware and software tests during boot time to verify the proper functioning of hardware and firmware. These tests validate the minimum set of operations necessary for the secure environment to operate correctly.
14. Covert Channel Analysis: Gemsos conducted an analysis of both storage-based and time-based covert channels using formal and informal methods. They employed techniques such as the Shared Resource Matrix to analyze covert channels and calculated their bandwidth using Shannon's method. The examination revealed 12 covert storage channels that could be closed with the appropriate system configuration outlined in the Trusted Facility Manual. Additionally, Gemsos identified various uncheckable covert time channels. To mitigate covert channels via semaphores, Gemsos uses Synchronizers and Counters for concurrency and deadlock handling.
15. Trusted Facility Management: GTNP does not support operator and administrator interfaces at runtime.
16. Trusted Recovery: GTNP ensures that system shutdown handles any component failures or interruptions caused by error diagnosis, non-recoverable interrupts, or the boot switch. It has mechanisms in place to restore a secure state upon reboot. Since GTNP does not trust any software outside of its own environment to enforce its security policy, failures elsewhere in the network do not impact GTNP.
17. Security Testing: The evaluation team conducted six weeks of functional and penetration tests on GTNP, categorized into device-specific, platform-specific, configuration-specific, and non-security-relevant tests. Five functional tests were performed, all of which passed without issues. Using the System Development Corporation (SDC) flaw hypothesis methodology, the team developed 47 infiltration hypotheses, of which 46 were examined. One software issue and four documentation issues were found and fixed, and two covert channels were investigated.
18. Design Specification and Verification: Gemsos used a formal model, FTLS, a DTLS, and mapped between the model and TCB. The formal language Ina Jo was used for the model. The correspondence across DTLS and the security model was specified. The FTLS described the hardware and security mechanisms of the kernel. A mapping of the policy model and

mechanisms, as well as the mechanisms and the kernel interface, was provided by Gemsos. Auditors reviewed the mappings and found no inaccuracies. They also reviewed the specifications of the GTNP and the GTNP kernel, finding no significant issues. Additionally, the specification to code mapping was reviewed and deemed satisfactory.

19. Configuration Management: Gemini implemented a Configuration Management plan throughout its Software Development Life Cycle (SDLC). This plan ensured control over all code related to security, testing and verification materials, hardware, and GTNP documentation. The Product Quality Assurance (QA) division and Configuration Control Board (CCB) were responsible for enforcing Gemini's CM procedures, with all divisions involved in hardware and software development participating in the CM strategy.

20. Trusted Distribution: Gemsos ships encrypted firmware and software. Altering the kernel software would require replacing the Gemini System Controller board, as the key management system uses system-specific keys burned into the System ID PROM, among other keys. Authenticators for the initial hardware delivery are securely shipped, preventing covert replacement of circuit boards or firmware.

21. Security Features User's Guide: Gemini developed a Security Features User's Guide (SFUG) for the Trusted Network Processor. Intended for programmers creating applications for the system, the guide provides a high-level view of the security features offered by the TCB, along with instructions on how to use and integrate them.

22. Trusted Facility Manual: Gemini provided a Trusted Facility Manual (TFM) for the assessment. The TFM includes instructions on setting up and configuring the system to deliver the evaluated configuration. It also covers system maintenance from the GTNP perspective after the system is operational, including validating trusted distribution.

23. Test Documentation: Gemsos conducted thorough testing and maintained comprehensive documentation. This included the types of tests performed, test case details, results, analysis of test outcomes, and patching of identified flaws. Test documentation covered various test categories, such as interface tests to verify TCB functionality, special engineering tests for critical features or subsystems, and niche scenario operations. Interface tests included exception, exception ordering, no-exception, and device testing. Unique engineering tests encompassed internal capability, memory management and such other testing mechanisms.

24. Design Documentation: Gemsos provided detailed design documentation, including system specifications for GTNP, software development and product specifications, interface requirements, code correspondence, covert channel analysis reports, and hardware reference manuals. Numerous other documents were also submitted by Gemsos.

25. RAMP: Gemsos had an NSA-approved Ratings Maintenance Plan (RM), outlining various policies to comply with updated TCSEC/TNI evaluation criteria. It specified personnel responsible for the RM plan and outlined procedures for re-verification when product changes occur.

As a result, Gemsos satisfied all 25 evaluation criteria and received an A1 rating. For hardware compatibility, it can run on various Gemini Trusted Base configurations, an IBM PC/AT configured for Gemini, or a Zenith Z-248. The Gemini Trusted Base system supports multilevel security and concurrent computing using 80286, 1386, or i486 microprocessors. Single 80286 microprocessor systems like the IBM PC/AT and Zenith Z-248 can also meet a range of computing needs. The Gemini System Controller board provides specialized support for GTNP, including the Data Ciphering Processor (DCP), read-only memory for encryption/decryption keys and system configuration, and an internal bus for transmitting keys to the DCP securely. Gemsos utilizes the hardware privilege levels available in the Intel 80x86 architecture. In GTNP, there are four hardware privilege levels, designated as PL0 to PL3, with PL0 being the highest privilege level and PL3 the lowest. In GTNP, all levels except PL0 are accessible outside the kernel [4][3.1] and the remaining levels are multiplexed into 8 [4][10.9] rings. The hardware of the GTNP supports two methods for transferring control from less-privileged subjects to more privileged subjects [4][3.2.1.1]: gates and traps. For memory management, GTNP uses segmentation which is supported by Intel 80x86 architecture. However, on certain processors such as i386 and i486 [4][3.2.3.1] and also GTNP uses paging. GTNP provides 8 gates (APIs) for management of segments.

Application Programming Interfaces (APIs)	Description
k_create_segment	Create a segment and add it to the naming hierarchy
k_terminate_segment	Remove a segment from a process's address space
k_flush_segment	Update the image of a segment on secondary storage
k_swapin_segment	Load a specific segment into a process's memory space
k_make_gate	Create a call gate for subjects in different rings
k_delete_segment	Remove a segment from secondary storage and the naming hierarchy
k_makeknown_segment	Map an existing segment (from the naming hierarchy) into a process's address space

In the mentioned APIs, segments are identified by their PLSN. Comparable gates are also present for volume management, process management, process synchronization, and I/O management.

GTNP's security-focused system design is among its most appealing features, as evidenced by its A1 rating, indicating high assurance and immunity to non-physical threats like Trojan horses and viruses, assuming it's used with proper security measures. This rating suggests that the system is likely free of exploitable bugs. It can operate on standard Intel 8086 hardware, making it compatible with modern commodity machines. Its primary application lies in sensitive environments such as government agencies, military installations, and intelligence services, where data confidentiality and integrity are paramount. However, it is not suitable for commercial use in end-user or corporate environments due to speed, performance, and software compatibility issues. Developed in Pascal, GTNP lags behind modern operating systems like Windows, MacOS, and GNU/Linux, which are predominantly written in C/C++ and Java, in terms of speed and performance. Despite the numerous security vulnerabilities in current operating systems, the prevailing technology and market trends prioritize performance over security. This preference, coupled with GTNP's slower speed resulting from its security-driven design in Pascal, makes it less appealing for widespread adoption. Furthermore, GTNP's kernel differs significantly from mainstream operating systems in terms of system calls and architecture, making applications developed for Windows, MacOS, and GNU/Linux incompatible without significant modifications. However, due to the lack of demand for GTNP-compatible applications, developers are unlikely to invest resources in adapting their software for GTNP. Consequently, despite its security advantages, GTNP's limitations in terms of performance, speed, and compatibility have hindered its adoption in the present day.

VI. Literature of Integrity-178B

Integrity-178B, developed by Green Hills Software, is an object-based operating system centered around a separation kernel. Like other separation kernels, it serves as a foundation for partitioning and hosting custom applications. The system ensures strict partition separation, allowing interactions only through predefined rules set by System Architects. Integrity-178B regulates access to different elements of the system such as memory, its devices, communication channels, and processing resources to maintain this separation. It is primarily designed for real-time embedded applications, lacking features like a file system, shell prompt, or user logins. The system schedules partitions to run on actual hardware and provides granular scheduling capabilities within each partition for tasks or entities. Due to its embedded nature, Integrity-178B cannot run on general-purpose hardware such as Intel 8086 or AMD processors.

Rated EAL 6+ with High Robustness according to SKPP standards, Integrity-178B requires a static configuration file during boot-up. This file contains access control policies for each partition, as well as the allocation of subjects and objects to partitions, and inter-partition flow policies. Each of the partitions provides an environment for multi-tasking applications. Security mechanisms are in place to prevent unauthorized access to the configuration file, as compromising it could lead to system subversion. Applications interact with the kernel and other partitions' applications [5][2.1] through a well-defined kernel API.

VII. Comparison of Integrity-178B's security functions to TCSEC evaluation criteria

The documentation for Integrity-178B largely aligns with SKPP regarding security requirements implementation. However, unlike GTNP, Integrity-178B does not provide detailed descriptions of its implementation methods; instead, it appears to restate SKPP's expectations. Therefore, when comparing Integrity-178B to TCSEC evaluation criteria, I will only mention the security requirement name and its corresponding number if it meets TCSEC's criteria, as there is little additional information about its specific implementation in Integrity-178B.

1. Object Reuse: Conforms to FDP_RIP.2: Full Residual Information Protection [6][5.1.2.4]
2. Labels: While MAC labels are not explicitly mentioned, equivalence classes fulfill a similar role and can be configured to function as MAC labels.
3. Label Integrity: Addressed indirectly through ADV_CTD_EXP.1: Configuration Tool Design [6][5.2.3.2] and AGD_ADM_EXP.1: Administrator Guidance [6][5.2.4.1]
4. Exportation of Labeled Information: Requirements not met
5. Exportation to Multilevel devices: Requirements not met
6. Exportation to Single-level devices: Requirements not met
7. Labeling Human-Readable Output: Partially addressed by ADV_CTD_EXP.1.3D: Configuration Tool Design [6][5.2.3.2], which specifies printing the initial system configuration vector in a human-readable format.
8. Subject Sensitivity Levels: Requirements not met
9. Device Labels: Requirements not met
10. Mandatory Access Control: Requirements not met
11. Trusted Path: Requirements not met
12. System Architecture: Addressed by ADV_ARC_EXP.1: Architectural Design [6][5.2.3.1] and ADV_INT_EXP.3: Minimization of Complexity [6][5.2.3.7]
13. System Integrity: Addressed by FAU_ARP.1: Security Alarms [6][6.1.1.4]; FMT_MOF.1: Security Management Functions [6][6.1.4.1]; FMT_MSA_EXP.1: Management of Security Attributes; FPT_TST_EXP: TSF Testing.
14. Covert Channel Analysis: Addressed by FDP_IFF.3: Limited Illicit Information Flows and AVA_CCA_EXP.2: Systematic Covert Channel Analysis.
15. Trusted Facility Management: Addressed by FMT_MOF.1: Management of Security Functions; FMT_MSA_EXP.1: Management of Security Attributes; FMT_MSA_EXP.3: Static Policy Attribute Initialization; FMT_MTD.1: Management of TSF Data; FMT_MTD.3: Secure TSF Data; FMT_MCD_EXP.1: Management of Configuration Data; and FMT_SMF.1: Specification of Management Function.

16. Trusted Recovery: Addressed by FPT_FLS.1: Failure with Preservation of Secure State; FPT_MTN_EXP.1; TOE Maintenance; FPT_MTN_EXP.2: TOE Maintenance Secure; FPT_RCV.4: Function Recovery; and FPT_RCV_EXP.2: Automated Recovery.
17. Security Testing: Addressed by ATE_COV.3: Rigorous Analysis of Coverage, ATE_DPT.3: Testing: Implementation Representation, ATE_FUN.2: Ordered Functional Testing, ATE_IND.3: Independent Testing – Complete.
18. Design Specification and Verification: Addressed by ADV_FSP_EXP.4: Formal Functional Specification; ADV_HLD_EXP.4: Semi Formal High-level explanation; ADV_RCR.3: Formal Correspondence; ADV_SPM.3: Formal TOE Security Policy Model.
19. Configuration management: Addressed by ACM_AUT.2: Complete CM Automation; ACM_CAP.5: Advanced Support; and ACM_SCP.3: Development Tools CM Coverage.
20. Trusted Distribution: Addressed by ADO_DEL_EXP.2: Detection of Modification
21. Security Features User's Guide: Addressed by AGD_USR.1: User Guidance
22. Trusted Facility Manual: Addressed by AGD_ADM_EXP.1: Administrator Guidance
23. Test Documentation: Addressed by ATE_COV.3: Rigorous Analysis of Coverage; ATE_DPT.3: Testing: Implementation Representation; ATE_FUN.2: Ordered Functional Testing; ATE_IND.3: Independent Testing
24. Design Documentation: Addressed by ADV_ARC_EXP.1, ADV_FSP_EXP.4, ADV_HLD_EXP.4, ADV_INI_EXP.1, ADV_INT_EXP.3, ADV_LLD_EXP.2, ADV_RCR.3, ADV_SPM.3, APT_PDF_EXP.1, APT_PSP_EXP.1, AVA_SOF.1, APT_PSP_EXP.1, and ADV_IMP_EXP.3; collectively satisfying TCSEC's design documentation criteria
25. RAMP: Addressed by AMA_AMP_EXP.1: Assurance Maintenance Plan

Therefore, Integrity-178B falls short of meeting all 25 TCSEC criteria, making it ineligible for an A1 rating according to TCSEC standards. A critical omission is its lack of Mandatory Access Control (MAC) implementation, a significant security gap. While Integrity-178B fulfills the 6 categories of requirements of security functions and the 9 security assurance requirements specified by SKPP, meeting these criteria only resulted in a high robustness rating, not an A1 rating as per TCSEC standards. However, Integrity-178B adheres to important security principles, including the principle of least privilege, kernel modularization and layering, kernel minimization, and trusted delivery. To prevent covert channels, Integrity-178B has intentionally disabled the L2 cache. Enabling this cache could potentially open up covert channel attack vectors through the L2 cache [5][11]. Given that Integrity-178B is designed for embedded systems, it may not be necessary for it to meet all 25 of TCSEC's criteria, as the real-world applications of Integrity-178B may differ from the specific security kernel requirements outlined by TCSEC.

VIII. Conclusion

TCSEC's 25 security kernel evaluation criteria are precise and stringent, requiring mandatory MAC implementation along with the use of segmentation, data hiding, layering, and various software engineering techniques to minimize the kernel and ensure verifiability. These criteria are crucial for building a secure operating system, aiming to prevent subversion at every stage of the software development life cycle (SDLC). They cover a wide range of aspects, including hardware, software, system resources, subjects, access policies, covert channels, documentation, software distribution, formal models, security proofs, design documentation, and testing documentation. Compliance with these standards, as demonstrated by GTNP, results in a secure system. GTNP has successfully met all of TCSEC's criteria, earning it an A1 rating. The system uses segments and employs a layered kernel design, with each layer being a Parnas module. It utilizes hardware privilege levels and implements MAC (BLP and Biba). GTNP also maintains comprehensive documentation, including a Formal Top-Level Specifications, FSPM, Descriptive top-level spec, covert analysis documentation, and testing documentation. The system effectively mitigates nearly all potential covert channels, for example, by using synchronizers and counters instead of semaphores. It fully complies with all other TCSEC criteria, including trusted distribution to prevent tampering during shipment to consumers. While GTNP is compatible with contemporary hardware that supports segmentation and hardware privilege levels, its performance is slower compared to modern operating systems like Windows and Linux. This limitation is partly due to GTNP's development in Pascal, which is type-safe and memory-safe, unlike modern OS developed in C/C++. Additionally, most modern applications are designed for compatibility with modern OS, not GTNP. GTNP is well-suited for environments prioritizing confidentiality and integrity, such as government agencies and military institutions, where security outweighs performance considerations. However, even in these environments, users require applications like text processors, spreadsheets, and web browsers, which may not be compatible with Gemsos. Thus, Gemsos may have limited applications even in these sectors. SKPP, a protection profile for separation kernels, offers an alternative to security kernels by dividing the system into partitions and allocating subjects and objects to each partition. Each partition may have an equivalence class, akin to a MAC label. However, upon comparing SKPP to TCSEC criteria, it is apparent that SKPP falls short, satisfying only 17 of the TCSEC criteria. Notably, SKPP lacks enforcement of MAC on the separation kernel, a significant security flaw, as MAC models like BLP and Biba are known for their security. Even for the criteria that SKPP does satisfy compared to TCSEC, it appears to dilute them, indicating a less stringent approach that could lead to security vulnerabilities. Nevertheless, SKPP includes additional security requirements to address threats such as environmental, platform, and separation-kernel specific threats, like unauthorized access to configuration data controlling partition flow. SKPP categorizes its security requirements into two parts: functional requirements, which must be met by every separation kernel, and assurance requirements, which a separation kernel must satisfy to achieve certification as a highly robust system. Integrity-178B, an embedded system OS based on a separation kernel, is certified as highly robust (EAL 6+). It meets all SKPP requirements, but falls short in comparison to TCSEC, satisfying only 17 criteria. Its most significant shortfall is the lack of mandatory access control (MAC), leaving the choice of security model to the user. Consequently, it does not qualify as an A1 system under TCSEC. The OS is tailored for embedded systems and is incompatible with standard Intel and AMD hardware, limiting its applications to specialized fields such as avionics. However, Integrity-178B adheres to key security principles such as least privilege, kernel modularization, and trusted delivery. It incorporates additional security measures not specified in TCSEC to address authentication, identification, and various threats. Integrity-178B has undergone covert channel analysis and disabled the L2 cache to prevent covert channel attacks. While not as secure as GTNP, Integrity-178B is moderately secure and has found applications in avionics. It includes sufficient security mechanisms to protect its static configuration file from unauthorized access.

IX. References

[1] An extract from the Trusted Network Interpretation (TNI) of the "Security Requirements for a Class A1 M-Component".
[2] Final Evaluation Report, Gemini Computers, Incorporated, Gemini Trusted Network Processor, National Computer Security Center, 28 June 1995.
[3] Common Criteria Evaluation and Validation Scheme Validation Report "Green Hills Software IN-ICR750-0402-GH01_Rel INTEGRITY-178B Separation Kernel", version 0.5, 31 January 2011.
[4] U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness (SKPP), Version 1.03, 29 June 2007.
[5] Green Hills Software INTEGRITY-178B Separation Kernel Security Target, Version 4.2, May 31, 2010.