TOE 설계문서 ver1.0

(ADV_TDS.5)

- 고등급 보안 마이크로커널 개발 -

<목 차>

1. 개요

본 문서는 TOE에 해당하는 CHAOS(Chibi-bsed High Assurance Operating System)의 내부 구조에 대해 설명한다.

1장은 본 문서의 개요와 목적을 보여주며 2장에서는 TOE의 내부 구조를 표현할 때, 활용되는 준정형 기법의 표기법에 대해 설명한다. 마지막으로 3장에서는 2장에서 설명한 표기법을 활용하여 TOE의 내부 구조를 서브시스템/모듈 수준으로 설명한다.

1.1. 문서의 목적

본 문서의 목적은 TOE 설계 내 보안기능에 보안기능 요구사항이 어떻게 적용되었는지 확인할 수 있는 충분한 정보를 정확하게 제공하는 것이다. 이때 평가자나 제 3자로 인해 TOE의 내부 구조가 부정확하게 이해되는 것을 방지하기 위해, 본 문서는 UML 중 Component Diagram을 활용하여 TOE의 내부 구조를 서브시스템/모듈 수준으로 설명한다.

2. Unified Modeling Language (UML)

본 장에서는 TOE의 내부 구조를 표현하기 위해 사용되는 표기법인 UML에 대해 설명한다. UML은 국제 표준화 기구 중 하나인 Object Management Group(이하 OMG)에서 제정한 통합 모델링 언어이다. 해당 언어는 활용 목적에 따라 12개의 다이어그램을 작성할 수 있다. 그중 본 문서 내 TOE의 설계는 서브시스템- 모듈간 세부 구성요소를 표현할 수 있는 컴포넌트 다이어그램을 활용하였다.

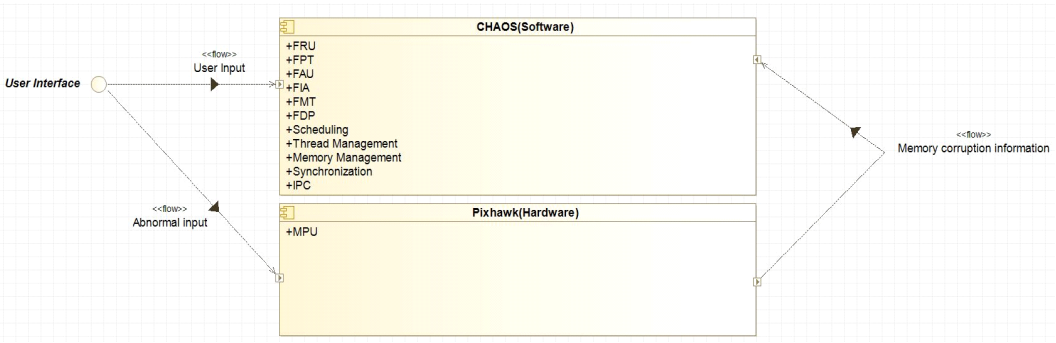
다이어그램 명	활용 목적
클래스 다이어그램	시스템의 구조를 클래스 관점에서 모델링
오브젝트 다이어그램	특정 시점에서의 시스템 동작과 클래스 간 관계를 모델링
컴포넌트 다이어그램	시스템의 기능을 컴포넌트(1개 이상의 클래스 포함) 관점에서 모델링
복합 구조 다이어그램	컴포넌트 간 계층 관계와 상호작용을 모델링
유즈케이스 다이어그램	사용자가 활용할 수 있는 시스템의 기능을 모델링
액티비티 다이어그램	시스템 실행 흐름을 모델링
시퀀스 다이어그램	객체 간 주고받는 메시지 교환을 시간의 흐름에 따라 모델링
배포 다이어그램	시스템이 실제 하드웨어에 탑재된 후, 실행되는 흐름을 모델링
패키지 다이어그램	UML 내 다양한 요소를 목적에 따라 그룹화하여 모델링
타이밍 다이어그램	특정 시간에서의 개체 동작이나 오브젝트 상태를 모델링
상태기계 다이어그램	입력 값이나 외부 이벤트에 따라 변경될 수 있는 시스템 상태를 모델링
통신 다이어그램	시스템이나 객체 간 상호작용을 모델링

[표 1] 위협과 보안목적의 매핑

3. TOE 설계

3.1. 서브시스템

본 장에서는 먼저 TSFI를 구성하는 TOE의 각 시스템 아키텍처, 서브시스템, 모듈 수준의 설계에 대한 간략한 설명을 제공한다. TOE가 탑재된 드론 시스템은 아래 그림과 같이 TOE에 해당하는 소프트웨어 형태의 CHAOS와 해당 TOE를 운영하기 위한 환경에 해당되는 하드웨어 보드인 Pixhawk로 구분된다. CHAOS 내부는 크게 SFR-수행 서브시스템에 해당하는 FRU, FPT, FAU, FIA, FMT, FDP와 SFR-지원 서브시스템인 Scheduling, Thread Management, Memory Management, Synchronization, IPC로 설계되었다. 이에 대한 컴포넌트 다이어그램은 아래와 같다.



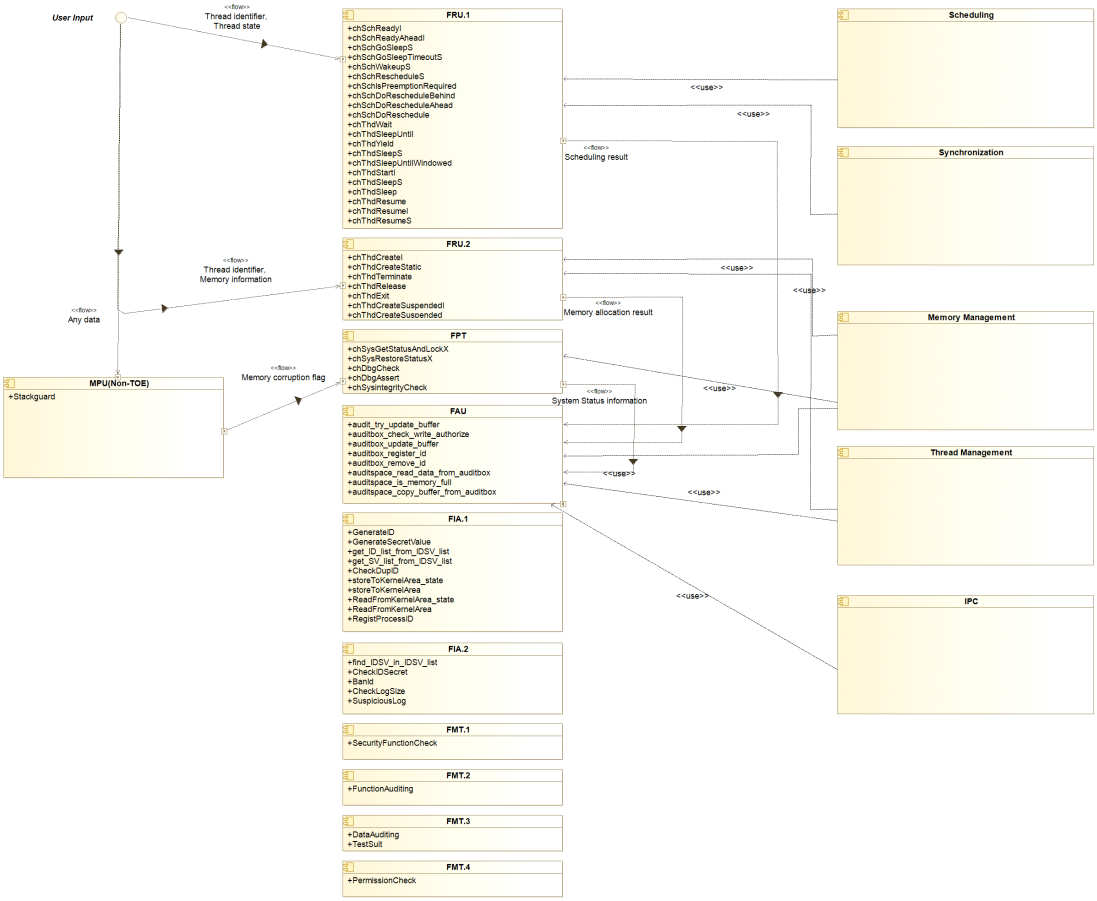
[그림 1] TOE가 탑재된 드론 시스템 내 서브시스템

서브시스템 명	상세 설명
FAU	커널이 동작하면서 발생하는 모든 이벤트를 감사 증적소에 저장하고 관리하는 기능을 수행하는 서브시스템
FRU	커널이 우선순위에 따라 사용자에게 자원을 할당하거나 회수하는 기능을 수행하는 서브시스템
FPT	커널이 외부 입력으로부터 자신을 항상 안전하게 보호하는 기능을 수행하는 서브시스템
FIA	커널에서 사용자 스레드의 생성, 실행 및 기타 기능에서의 스레드의 식별 및 인증하는 기능을 수행하는 서브시스템
FMT	커널에 추가된 보안기능의 무결성을 검증하고 보안기능의 동작을 기록하는 기능을 수행하는 서브시스템
FDP	커널이 관리하는 시스템 자원을 인가되지 않은 사용자 스레드의 접근으로부터 보호하는 기능을 수행하는 서브시스템

[표 1] 고등급 보안 마이크로커널 서브시스템

3.2. 모듈

TOE 설계 내 서브시스템은 설계에 적용하고자 하는 보안기능 요구사항과 보호하고자 하는 자산에 따라 FRU.1, FRU.2과 같은 모듈로 구분될 수 있다. 서브시스템 내 모듈은 소스 코드 내 수행 기능(함수 단위)의 목적에 따라 구분되며, 모듈을 구성하는 TSFI(시스템 함수)는 호출 조건에 따라 구분된다.



[그림 2] TOE가 탑재된 드론 시스템 내 모듈

3.2.1. FAU

본 절에서는 (준)정형하게 설계된 FAU 모듈에 수행 목적과 수행 동작을 기술한다.

모듈 명	세부 기능 명	수행 목적
FAU	init_mailbox	사용자 스레드가 전송하는 감사 이벤트 정보를 저장하기 위한 영역과 그에 대한 접근 권한 정의되는 데이터 구조체인 mailbox 초기화
	scan_access_thread	감사 증적소에 접근할 수 있는 스레드의 식별자 값을 확인하기 위해 현재 ready list 검색

FAU	update_mailbox_accessible_thread	함수를 호출한 후, 감사 증적소에 접근할 수 있는 스레드 식별자를 메일박스에 입력
	check_validity	임의의 스레드가 감사 증적소에 접근하는 경우, 해당 스레드의 식별자가 mailbox 내 저장된 스레드 식별자와 일치하는지 점검
	scan_permit_thread	mailbox 내 스레드 식별자 리스트 내 원소를 하나씩 정수형으로 개별 변환
	copy_wr_thread: scan_permit_thread	함수를 통해 식별한 스레드 식별자 목록을 생성
	copy_buffer	감사 내용을 증적소에 저장하기 위해 해당 값을 mailbox에 저장
	audit_data_read	mailbox에 저장할 감사 기록을 임시 메모리 공간에 저장
	mailbox_write_authorize	mailbox에 접근한 스레드에게 쓰기 권한을 인가
	update_mailbox_audit_data	감사 기록을 mailbox 내 기록
	update_mailbox_buffer	타 스레드가 접근할 수 있도록 감사 증적소에 대한 접근 플래그를 변경
	storage_init	감사 기록이 저장될 증적소 메모리 크기를 할당
	storage_overwrite	증적소가 포화되는 경우, 가장 오래된 감사 기록대시 새로운 감사기록을 저장
	storage_append	증적소 내 잔여 메모리 공간에 감사 기록을 저장
	memory_increment	증적소의 잔여 메모리 공간을 계산
	memory_status_check	적소 내 잔여 메모리 공간을 확인한 후, 알람 메시지를 출력

[표 2] FAU 설계 목적

상기 목적을 만족하기 위한 FAU 모듈 내 각 TSFI의 동작은 (준)정형기법 도구인 Isabelle/HOL을 활용하여 아래와 같이 설계하였다.

모듈명	세부 기능 명	설계 및 검증 내역
FAU	init_mailbox	text‹initialize mailbox› definition init_mailbox::"new_mailbox" where "init_mailbox = ⦇buffer = init_share_space, boundary = (0,256), wr_thread = [], flag = False⦈"
	scan_access_thread	text‹scan ready list› primrec scan_access_thread::"ready_list⇒nat list⇒nat list" where "scan_access_thread [] ns = []" | "scan_access_thread (x # xs) ns = scan_access_thread xs ns @ [fst x]"
	update_mailbox_accessible_thread	text‹update accessible thread list ns = wr_thread› definition update_mailbox_accessible_thread::"ready_list⇒nat list⇒new_mailbox" where "update_mailbox_accessible_thread rls ns = (if (length rls) = 0 then init_mailbox else ⦇ buffer = init_share_space, boundary = (0, 128), wr_thread = scan_access_thread rls ns, re_thread = 0, flag = False⦈)"
	scan_permit_thread	text‹scan ready list› primrec scan_permit_thread::"nat list⇒nat list" where "scan_permit_thread [] = []" | "scan_permit_thread (x # xs) = [x] @ scan_permit_thread xs"
	copy_wr_thread	text‹copy new nat list for wr_thread› definition copy_wr_thread::"new_mailbox⇒nat list" where "copy_wr_thread nm = (if(wr_thread nm = []) then [] else scan_permit_thread (wr_thread nm))"
	check_validity	text‹check thread access is valid › primrec check_validity::"nat list⇒nat⇒result" where "check_validity [] n = Fail" | "check_validity (x # xs) n = (if x = n then Success else check_validity xs n)"

FAU	mailbox_write_authorize	text‹check thread access and reply with flag› definition mailbox_write_authorize::"new_mailbox⇒thread⇒thread × result" where "mailbox_write_authorize nm t = (if (check_validity (copy_wr_thread nm) (fst t)) = Success then (t,Success) else (t, Fail))"
	copy_buffer	text‹copy record for buffer› definition copy_buffer::"new_mailbox⇒audit_data" where "copy_buffer nm = (if buffer nm = init_share_space then init_share_space else ⦇id = (id(buffer nm)), event = (event(buffer nm)), status = (status(buffer nm))⦈)"
	audit_data_read	text‹message read› definition audit_data_read::"new_mailbox⇒thread⇒audit_data × result" where "audit_data_read nm t = (if ((re_thread nm) = (fst t)) then (copy_buffer nm, Success) else (copy_buffer nm, Fail))"
	update_mailbox_ audit_data	text‹audit data update› definition update_mailbox_audit_data::"audit_data⇒audit_data" where "update_mailbox_audit_data au = ⦇id = id au ,event = event au, status = status au⦈"
	update_mailbox_ buffer	text‹audit data buffer update› definition update_mailbox_buffer::"new_mailbox⇒audit_data⇒new_mailbox" where "update_mailbox_buffer nm au = ⦇buffer = au, boundary = boundary nm, wr_thread = wr_thread nm, re_thread = re_thread nm, flag = True⦈"
	audit_data_write	text‹message write› definition audit_data_write::"new_mailbox⇒thread⇒audit_data⇒new_mailbox × result" where "audit_data_write nm t au = (if ((snd (mailbox_write_authorize nm t)) = Success) then (update_mailbox_buffer nm au, Success) else (nm, Fail))"
	storage_init	text‹Set audit storage(memory) initialization› definition storage_init::"audit_storage" where "storage_init = (0, 1024)"
	storage_ overwrite	text‹audit-overwrite› definition storage_overwrite::"audit_storage⇒audit_storage" where "storage_overwrite as = (16-((snd as - fst as)), snd as)"
	storage_ append	text‹audit-append› definition storage_append::"audit_storage⇒audit_storage" where "storage_append as = ((fst as) + 16, snd as)"
	memory_increment	text‹memory increment status› definition memory_increment::"audit_storage⇒audit_storage" where "memory_increment as = (if snd as -fst as ≥ 16 then storage_append as else storage_overwrite as)"
	memory_status_check	text‹memory alarm› definition memory_status_check::"audit_storage⇒FAU" where "memory_status_check as = (if fst as ≤ 0 then Audit_Storage_is_Full else if (fst as < 871) ∧ (fst as > 0) then Audit_Storage_is_remained_enough else Audit_Storage_is_85_percent_full)"

[표 3] FAU 세부 설계 내역

3.2.2. FRU

본 절에서는 (준)정형하게 설계된 FRU 서브시스템의 세부 모듈에 대한 수행 목적과 수행 동작을 기술한다.

모듈 명	세부 기능 명	수행 목적
FRU.1	Thread_Reschedule	우선순위가 가장 높은 순위를 가지는 스레드에게 자원 분배
	Thread_Execute	스레드 실행
	Thread_Suspend	자신보다 높은 우선순위를 가지는 스레드에게 자원 양도
	Thread_Resume	스레드 재실행
FRU.2	Thread_Create	스레드가 요청한 만큼 메모리 할당
	Thread_Delete	스레드에게 요청한 메모리 할당 해제

[표 4] FRU 설계 목적

설계된 모듈은 수행목적과 보호하고자 하는 TSF 데이터에 따라 다음과 같이 두 가지로 구분된다. FRU.1은 우선 순위에 따라 할당된 시간동안 스레드에 자원을 분배하기 위한 스케줄링 기능을 수행하기 위해 설계된 모듈이다. FRU.2는 요청된 크기만큼 메모리를 할당하기 위해 설계된 모듈이다. 이에 대한 상세 내역은 아래와 같다.

모듈 명	세부 기능 명	수행 목적

FRU.1	Thread_Reschedule	definition chThdCreate :: "thread_descriptor_t ⇒ SecretValue ⇒ (unit, thread_t) state"
		where "chThdCreate tdp kValue =
		(
		if kValue = KernelSecret then
		let registeredIDSV = RegistProcessID kValue in
		if snd registeredIDSV then do{
		(if CH_CFG_USE_REGISTRY ∧ (CH_DBG_ENABLE_STACK_CHECK ∨ CH_CFG_USE_DYNAMIC) then do{
		let tmp = chDbgAssert ((chRegFindThreadByWorkingArea (wbase tdp)) = NullThread) ''working area in use'';
		return (NullThread)
		}
		else
		return (NullThread)
		);
		(if CH_DBG_FILL_THREADS then do {
		let tmp = thread_memfill (wbase tdp) (wend tdp) CH_DBG_STACK_FILL_VALUE;
		return (NullThread)
		}
		else
		return (NullThread)
		);
		(do{
		let tmp = chSysLock;
		let tp = snd (run_state (chThdCreateSuspendedI tdp kValue) NullThread);
		let tmp = chSchWakeupS tp MSG_OK;
		let tmp = chSysUnlock;
		return (tp)
		})
		}
		else
		return (NullThread)
		else
		return (NullThread)
		)"

	Thread_Execute	definition chThdTerminate :: "SecretValue ⇒ ch_threads ⇒ (kernel, unit) state"
		where "chThdTerminate kSec tp = do{
		chSysLock;
		
		curr_thd <- getThread_id tp;
		curr_thd_pass <- getThread_pass tp;
		CheckIDSecret curr_thd curr_thd_pass;
		verify_result <- getVerification;
		
		(if verify_result
		then do {
		let curr_thd_flag = flags tp;
		let tmp = bitCalculation curr_thd_flag CH_FLAG_MODE_TERMINATE bitOr;
		tp <- setFlagsThreads CH_FLAG_MODE_TERMINATE;
		chSysUnlock
		}
		else
		chSysUnlock)
		}"

	Thread_Suspend	definition chSchRescheduleS :: "(kernel, unit) state" where
		"chSchRescheduleS = do {
		k <- get_kernel;
		if (chSchIsRescRequiredI k)
		then chSchDoRescheduleAhead
		else return ()
		}"

	Thread_Resume	definition chThdStart :: "SecretValue ⇒ ch_threads ⇒ ID ⇒ SecretValue ⇒ (thread_t, unit) state"
		where "chThdStart kValue tp id_in sv_in = do {
		let tmp = chSysLock;
		(if kValue = KernelSecret then
		(if CheckIDSecret kValue id_in sv_in then do {
		let tmp = chDbgAssert ((state_to_value (state tp)) = CH_STATE_WTSTART) ''wrong state'';
		let tmp = chSchWakeupS tp MSG_OK;
		let tmp = chSysUnlock;
		return ()
		}
		else do{
		let tmp = chSysUnlock;
		return ()
		})
		else
		return ()
		)
		}"

<table>
<tr><td rowspan="2">FRU.2</td><td>Thread_Create</td><td>

```
definition chThdWait :: "SecretValue ⇒ ch_threads ⇒ (unit, msg_t) state"
  where "chThdWait kValue tp = do{
    let tmpBool = chDbgCheck (tp ≠ NullThread);
    let currp = currp ch;
    let tmp = chSysLock;
    let curr_thd_id = id_thd tp;
    let curr_thd_pass = pass_thd tp;
    (if CheckIDSecret kValue curr_thd_id curr_thd_pass then do{
      let tmp = chDbgAssert (tp ≠ currp) ''waiting self'';
      (if CH_CFG_USE_REGISTRY then do{
        let tmp = chDbgAssert (refs tp > 0) ''no references'';
        return ()
      } else
        return ()
      );
      (if (state tp ≠ (value_to_state CH_STATE_FINAL)) then do {
        let tmp = list_insert currp waiting_thread (first_pos (λx. (x=tp)) waiting_thread) (queue (rlist ch));
        let tmp = chSchGoSleepS CH_STATE_WTEXIT;
        return ()
      } else
        return ()
      );
      let msg = exitcode (u tp);
      chSysUnlock;
      (if CH_CFG_USE_REGISTRY then do {
        let tmp = chThdRelease tp;
        return ()
      } else
        return ()
      );
      return (msg)
    } else
      return (0)
    )
  }"
```

</td></tr>
<tr><td>Thread_Delete</td><td>

```
definition chThdResume :: "thread_reference_t ⇒ msg_t ⇒ (unit, unit) state"
  where "chThdResume trp msg_in = do{
  chSysLock;
  chThdResumeS trp msg_in;
  chSysUnlock
  }"
```

</td></tr>
</table>

[표 5] FRU 설계 목적

3.2.3. FPT

본 절에서는 (준)정형하게 설계된 FPT 서브시스템의 세부 모듈에 대한 수행 목적과 수행 동작을 기술한다.

모듈 명	세부 기능 명	수행 목적
FPT	ChSysRestoreStatusX	시스템이 불안전한 상태에 도달하는 경우에 복구하는데 활용될 수 있는 시스템 값 저장
	ChSysGetStatusX	시스템이 불안전한 상태에 도달하는 경우에 복구하는데 활용될 수 있는 시스템 값 호출
	IsintegrityCheck	시스템이 불안정한 상태에 빠졌는지 확인

[표 6] FPT 설계 목적

상기 목적을 만족하기 위한 FAU 모듈 내 각 TSFI의 동작은 (준)정형기법 도구인 Isabelle/HOL을 활용하여 아래와 같이 설계하였다.

모듈 명	세부 기능 명	설계 및 검증 내역

FPT	ChSysRestoreStatusX	`definition chSysRestoreStatusX :: "syssts_t ⇒ (unit, unit) state"`
		`  where "chSysRestoreStatusX sts_in =`
		`  (if port_irq_enabled sts_in then`
		`  (if port_is_isr_context then`
		`  chSysUnlockFromISR`
		`  else do{`
		`  chSchRescheduleS;`
		`  chSysUnlock`
		`  })`
		`  else`
		`  return()`
		`  )"`

	ChSysGetStatusX	definition chSysGetStatusAndLockX :: "(unit, syssts_t) state"
		where "chSysGetStatusAndLockX = do{
		let sts = port_get_irq_status;
		if port_irq_enabled sts then
		if port_is_isr_context then do {
		chSysLockFromISR;
		return sts
		}
		else do {
		chSysLock;
		return sts
		}
		else
		return sts
		}"

	IsintegrityCheck	definition chSysIntegrityCheckI :: "nat ⇒ nat ⇒ nat ⇒ nat ⇒ (unit, bool) state"
		where "chSysIntegrityCheckI testmask tp_index vtp_index rlist_list_index = do {
		return (chDbgCheckClassI);
		(if (bitCalculation testmask CH_INTEGRITY_RLIST bitAnd) ≠ 0 then
		do {
		let tp = (queue (rlist ch))!tp_index;
		let queue_tmp = queue (rlist ch);
		let first_queue = drop (tp_index+1) queue_tmp;
		let n_value_first = first_pos (λx. (x≠tp)) first_queue;
		let second_queue = rev (take tp_index queue_tmp);
		let n_value_second = first_pos (λx. (x≠tp)) second_queue;
		let n_value = n_value_first - n_value_second;
		(if n_value ≠ 0 then
		return (True)
		else
		return (False)
		)
		}
		else
		return (False)

[표 7] FPT 모듈 설계 내역

3.2.4. FIA

본 절에서는 (준)정형하게 설계된 FIA 서브시스템의 세부 모듈에 대한 수행 목적과 수행 동작을 기술한다.

모듈 명	세부 기능 명	수행 목적
FIA.1	RegistProcessID	인증에 필요한 식별자와 비밀 값을 생성
	GenerateID	
	GenerateSecretValue	
	StoreToKernelArea	
	CheckDupID	ID 중복 확인
	ReadFromKernelArea	
FIA.2	CheckIDSecret	인증
	SuspiciousLog	인증 횟수 제한
	CheckLogSize	
	BanId	인증 금지
	SendEmailtoAdmin	

[표 8] FIA 설계 목적

설계된 모듈은 수행목적과 보호하고자 하는 TSF 데이터에 따라 다음과 같이 두 가지로 구분된다. FIA.1은 쓰레드를 생성/실행하기 전에 커널에서 수행되어야하는 식별 절차를 수행하기 위해 설계된 모듈이다. FIA.2는 쓰레드를 실행하거나 쓰레드의 요청을 처리하기 전에 커널에서 수행되어야하는 인증 절차를 수행하기 위해 설계된 모듈이다. 이에 대한 상세 내역은 아래와 같다.

모듈 명	세부 기능 명	수행 목적
FIA.1	RegistProcessID	record IDSV = id::ID pass::SecretValue definition tempIDSV :: IDSV where "tempIDSV ≡ ⦇id = 1, pass = 123⦈" definition RegistProcessID:: "PID ⇒ ID ⇒ SecretValue ⇒ SecretValue ⇒ IDSV" where "RegistProcessID pid id_in pass_in kSecret ≡ (if pid = 0 then tempIDSV⦇id:=GenerateID id_in kSecret, pass:=GenerateSecretValue pass_in kSecret⦈ else tempIDSV)"

	GenerateID	definition GenerateID :: "SecretValue ⇒ ID × bool" where
		"GenerateID kValue =
		(
		if kValue = KernelSecret then
		((pseudo_random seedIDlist),True)
		else
		(seedIDlist,False)
		)"

	GenerateSecretValue	definition GenerateSecretValue :: "SecretValue ⇒ SecretValue × bool" where
		"GenerateSecretValue kValue =
		(
		if kValue = KernelSecret then
		((pseudo_random seedSVlist), True)
		else
		(seedSVlist, False)
		)"

FIA.1	StoreToKernelArea	definition storeToKernelArea :: "SecretValue ⇒ ID ⇒ SecretValue ⇒ bool"
		where "storeToKernelArea kValue id_in sv_in =
		(
		if fst (run_state (storeToKernelArea_state kValue id_in sv_in) IDSV_list) then
		True
		else
		False
		)"

FIA.1	CheckDupID	definition CheckDupID :: "ID ⇒ SecretValue ⇒ bool" where
		"CheckDupID id_in kValue =
		(
		if kValue = KernelSecret then
		if id_in ∉ set (get_ID_list_from_IDSV_list IDSV_list []) then
		True
		else
		False
		else
		False
		)"

	ReadFromKernelArea	definition getPID :: "Kernel ⇒ PID" where "getPID r ≡ pid r" definition getID :: "Kernel ⇒ ID" where "getID r ≡ id r" definition getSV :: "Kernel ⇒ SecretValue" where "getSV r ≡ pass r" definition ReadFromKernelArea :: "SecretValue ⇒ Kernel" where "ReadFromKernelArea kSecret ≡ (if kSecret = KernelSecret kSecret then tempK else emptyK)"
FIA.2	CheckIDSecret	definition CheckIDSecret :: "SecretValue ⇒ ID ⇒ SecretValue ⇒ bool" where "CheckIDSecret kSecret id_in sv_in ≡ (if kSecret = KernelSecret then find_IDSV_in_IDSV_list kSecret id_in sv_in IDSV_list else False)"
	SuspiciousLog	definition SuspiciousLog :: "SecretValue ⇒ ID ⇒ SecretValue ⇒ timestamp ⇒ nat ⇒ Logs ⇒ Logs" where "SuspiciousLog kSecret id_in SV_in ts_in trials log_in ≡ (if trials < 5 then setLog id_in ts_in else BanId kSecret id_in SV_in ts_in log_in)"

FIA.2	CheckLogSize	record sLog = filesize::Size definition CheckLogSize :: "filesize ⇒ bool" where "CheckLogSize sLog ≡ (if (sLog ≥ 9999) then False else True)" (* 9999 is MaxSize*)
	BanId	definition BanId :: "SecretValue ⇒ ID ⇒ SecretValue ⇒ timestamp ⇒ Logs ⇒ Logs" where "BanId kSecret id_in SV_in ts_in log_in ≡ (if CheckIDSecret kSecret id_in SV_in then log_in else SendEmailtoAdmin id_in ts_in log_in)"
	SendEmailtoAdmin	definition SendEmailtoAdmin :: "ID ⇒ timestamp ⇒ Logs ⇒ Logs" where "SendEmailtoAdmin id_in ts_in log_in ≡ (if sendEmail id_in ts_in = tempE then log_in else log_in)"

[표 9] FIA 모듈 설계 내역

3.2.5. FMT

본 절에서는 (준)정형하게 설계된 FMT 서브시스템의 세부 모듈에 대한 수행 목적과 수행 동작을 기술한다.

모듈 명	세부 기능 명	수행 목적
FMT.1	SecurityFunctionCheck	입력된 보안기능을 확인
FMT.2	FunctionAuditing	보안기능을 확인하고자 하는 함수의 행동을 기록
FMT.3	DataAuditing	데이터의 이름과 행동을 입력받아 허가된 데이터 행동에 포함되는지 판별
	TestSuit	테스트 수트를 입력받아 미리 선정한 테스트 수트에 포함되는지 여부 판별
FMT.4	PermissionCheck	메모리 주소를 입력받아 해당 주소의 권한이 적합한지 확인

[표 10] FMT 설계 목적

설계된 모듈은 수행목적과 보호하고자 하는 TSF 데이터에 따라 다음과 같이 두 가지로 구분된다. FMT.1은 보안기능이 들어간 함수들을 입력받아 보안기능이 정상적으로 들어갔는지 확인하기 위해 설계된 모듈이다. FMT.2는 보안기능이 들어간 함수들의 행동을 기록하기 위해 설계된 모듈이다. FMT.3은 데이터의 행동을 기록하기 위해 설계된 모듈이다. 마지막으로 FMT.4는 로그 기록 및 메모리 데이터에 접근 요청에 대한 권한을 확인하기 위해 설계된 모듈이다. 이에 대한 상세 내역은 아래와 같다.

모듈 명	세부 기능 명	수행 목적
FMT.1	SecurityFunctionCheck	definition SecurityFunCheck :: name_check where "SecurityFunCheck name pool ≡ (if name ∈ pool then True else False)"

FMT.2	FunctionAuditing	definition FunctionAuditing :: f_audit where "FunctionAuditing func_behavior normal_behavior≡ (if func_behavior ∈ normal_behavior then True else False)"
FMT.3	DataAuditing	definition dataAuditing :: d_audit where "dataAuditing data_name data_behavior ≡ (if data_name ∈ data_behavior then True else False)"
	TestSuit	definition TestSuit :: suite where "TestSuit name test ≡ (if name ∈ test then True else False)"
FMT.4	PermissionCheck	primrec check_permission :: "user ⇒ belong list ⇒ permission list" where init: "check_permission A [] = []" | data_check: "check_permission A (bl # bls) = (if (snd bl) = A then allow # check_permission A bls else deny # check_permission A bls) "

[표 11] FMT 설계 모듈 내역

3.2.6. FDP

본 절에서는 (준)정형하게 설계된 FDP 서브시스템의 세부 모듈에 대한 수행 목적과 수행 동작을 기술한다.

모듈 명	세부 기능 명	수행 목적
FDP.1	valOfSecurityProfile	입력받은 보안 속성이 커널의 보안속성인지 확인
FDP.2	VerifyDataflow	데이터의 권한과 접근하는 메모리 영역의 권한 확인
	ControlAccess	명령어 권한 확인
FDP.3	CryptoBufClear	암호화 버퍼의 초기화
	CMDbufClear	명령어 버퍼의 초기화

[표 12] FDP 설계 목적

설계된 모듈은 수행목적과 보호하고자 하는 TSF 데이터에 따라 다음과 같이 두 가지로 구분된다. FDP.1은 보안속성을 입력받아 보안속성을 확인하기 위한 목적을 만족하기 위해 설계된 모듈이다. FDP.2는 보안속성 및 정보 보안속성 유형에 기반하여 접근 권한 여부를 판별하기 위해 설계된 모듈이다. 마지막으로 FDP.3은 자원의 모든 이전 정보 내용을 초기화하기 위해 설계된 모듈이다. 이에 대한 상세 내역은 아래와 같다.

모듈 명	세부 기능 명	설계 및 검증 내역
FDP.1	valOfSecurityProfile	definition valOfSecurityProfile :: "string ⇒ bool" where "valOfSecurityProfile code = (if code = (security_profile ch) then True else False)"
FDP.2	VerifyDataflow	definition VerifyDataflow :: "data ⇒ address ⇒ bool" where "VerifyDataflow data_in address_in = (if permission_level data_in = permission_level (memory!address_in) then True else False)"

	ControlAccess	definition ControlAccess :: "instruction ⇒ thread_t ⇒ bool"
		where "ControlAccess com thd_in =
		(let com_pair = find_command (λx. x=com) command_list in
		if ((length command_list) ≠ com_pair) ∧
		(inst_perm_level (command_list!com_pair)) = (perm_thd thd_in) then
		True
		else
		False
		)"

FDP.3	CryptoBufClear	definition CryptoBufClear :: "memory_t" where "CryptoBufClear = append (append (take CryptoBufStart memory) (replicate (CryptoBufEnd-CryptoBufStart) (⦇val=0,permission_level=Kernel⦈::data))) (drop CryptoBufEnd memory)"
	CMDbufClear	definition CMDbufClear :: "memory_t" where "CMDbufClear = append (append (take CMDbufStart memory) (replicate (CMDbufEnd - CMDbufStart) (⦇val=0,permission_level=Kernel⦈::data))) (drop CMDbufEnd memory)"

[표 13] FDP 모듈 설계 내역