Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.29, NO.4, Aug. 2019 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2019.29.4.821

고신뢰 드론 시스템을 위한 스케줄링 측면에서의 서비스 거부 공격 완화 방안 연구*

곽 지 원,[1†] 강 수 영,[2] 김 승 주[2‡]
[1]고려대학교 일반대학원, [2]고려대학교 정보보호대학원

Study on Highly Reliable Drone System to Mitigate Denial of Service Attack in Terms of Scheduling*

Ji-Won Kwak,[1†] Soo-Young Kang,[2] Seung-Joo Kim[2‡]
[1]Graduate School, [2]Center for Information Security Technologies(CIST), Korea University

요 약

사이버 보안위협이 증가하면서 시스템 보안 수준이 보증된 고신뢰 시스템에 대한 수요가 증가하고 있다. 정보보호 제품에 대한 평가·인증 국제 표준인 공통평가기준(CC: Common Criteria)은 고등급 보안 수준을 보증하기 위해 시스템에 대한 정형명세 및 검증을 요구하고 있으며 점차 이를 준수하는 사례가 증가하고 있다. 본 논문에서는 고등급 보안 수준을 보증하기 위해 다양한 분야에 적용 가능하며 고신뢰 수준을 요구하는 드론시스템에 대한 보안 위협을 도출한다. 그 결과를 기반으로 시스템 커널 내 스케줄링 측면에서 개선된 시스템 모델을 Z/EVES를 활용하여 정형명세 및 검증을 진행함으로써 고신뢰 드론시스템에 적용 가능한 스케줄링 방식을 제안한다.

ABSTRACT

As cyber security threats increase, there is a growing demand for highly reliable systems. Common Criteria, an international standard for evaluating information security products, requires formal specification and verification of the system to ensure a high level of security, and more and more cases are being observed. In this paper, we propose highly reliable drone systems that ensure high level security level and trust. Based on the results, we use formal methods especially Z/EVES to improve the system model in terms of scheduling in the system kernel.

Keywords: High-Assurance, formal specification, verification, denial of service, drone system

I. 서 론

2018년 출판된 파이어아이의 보안 예측 보고서인 'Facing Forward: Cyber Security in 2019 and Beyond'에 따르면 국가 차원의 사이버 공격 위협이 증가하고 있다 [1]. 따라서 높은 보안 수준이 요구되는 군사용 시스템, 전기차, 스마트 팩토리 등의 CPS(Cyber-Physical System) 환경에서 높은 보안성 및 보증 수준을 확보하기 위한 시도가 이루어지고 있다. 실제로 미 국방부는 HACMS(High-Assurance Cyber Military Systems) 프로젝트를 통해 군사용 드론과 사이버무기를 포함한 운용 중인 정보시스템에 대한 보안성을 보증하기 위한 연구 활동을 지원하고 있다 [2]. 또한 HACMS 프로젝트를 통해 자국에서 조성한 고신뢰

Received(06. 03. 2019), Modified(07. 01. 2019), Accepted(07. 02. 2019)
* 이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2018-0-00532,고등급(EAL6 이상) 보안 마이크로 커널 개발)
† 주저자, jwkwak4031@korea.ac.kr
‡ 교신저자, skim@korea.ac.kr(Corresponding author)

Fig. 1. Darpa HACMS Project

시스템 프레임워크를 확장시키기 위해 LandShark UGV, ArduPilot, AR-Drone 과 같은 오픈소스를 활용하여 프로젝트를 진행하고 있다.

HACMS 프로젝트에서 군사용 드론에 대한 보증을 수행하기 위해 활용 중인 오픈소스인 Ardupilot을 선정하였다. Ardupilot은 ChibiOS 마이크로커널을 사용하며 해당 커널은 ARM Cortex M3 뿐만 아니라 STM 시리즈를 비롯한 다양한 하드웨어 플랫폼에 탑재될 수 있다는 확장성 때문에 다양한 드론 개발 커뮤니티를 통해서 빠르게 사회 전반에 확산되고 있다. 따라서 해당 커널이 탑재된 드론에 대해 보안 사고가 발생할 경우 시장에 큰 파장을 불러일으킬 수 있다.

정보시스템의 보안 및 보증 수준을 평가하기 위한 대표적인 국제 기준으로 공통평가기준이 있다. 1998년 국제 표준으로 제정된 공통평가기준은 시스템이 높은 수준의 평가등급(EAL: Evaluation Assurance Level)을 확보하기 위해 Z, B, Isabelle/HOL 등과 같은 정형기법을 활용하여 시스템 동작을 명세 및 검증하도록 요구하고 있다 [3].

정형기법은 크게 정형명세(formal specification)와 정형검증(formal verification)으로 나뉜다. 정형명세는 개발하고자하는 시스템의 요구사항을 수학적 기호로 이루어진 식이나 논리 기호로 기술하는 것이다. 정형명세에 사용되는 언어는 명제의 내부를 분석하여 기호화하는 술어논리 중 원소에만 한정사를 가하는 일차논리(FOL: First Order Logic) 기반 언어인 Z, B와 고차논리(HOL: High Order Logic) 기반의 HOL, Coq 언어 등이 존재한다. 정형검증의 경우 상기 언어로 명세된 시스템 모델을 수학적 추론을 통해 검증함으로써 요구사항이 올바르게 반영되었는지 증명하는 것이다. 이를 위해 사용되는 도구는 명세 언어에 따라 다르지만 많은 연구가 Z/EVES, ProB, Coq Proof Assistant, Isabelle/HOL 등을 통해 수행되고 있다 [4, 5].

따라서 본 논문에서는 ChibiOS가 탑재된 드론 시스템에 대한 보안성을 분석하고 식별된 보안 위협 가운데 서비스 거부공격 일부가 개선된 모델에 대해 정형 명세 및 검증한다.

서론에 이어 2장에서는 드론과 같은 항공시스템에 대한 정형 검증을 수행한 연구를 소개하고, 3장에서는 보안위협모델링을 실시하기 위한 군사용 드론의 공통적인 시스템 모델을 도출한다. 이후 4장에서는 보안위협모델링을 통한 보안성 분석 결과를 제시한다. 5장에서는 분석 결과를 토대로 스케줄링 측면에서 개선된 정형화된 시스템 모델을 제시하고, 6장에서는 해당 모델에 대한 정형 검증 결과를 제시한다.

II. 관련 연구

1980년 초기, 시스템 개발은 대부분 한글, 영어와 같이 인간이 일상생활에서 사용하는 비정형 언어로 요구사항을 명세하고 이를 기반으로 개발된 시스템을 시뮬레이션하는 것이 주를 이루었다. 하지만 Ariane 5, Mars Pathfinder와 같이 시뮬레이션 방식으로 점검된 시스템이 개발자가 의도하지 않은 내부 오류로 인해 통제 불능 상태에 빠질 수 있다는 것이 밝혀지면서 정형검증 연구가 활발해지기 시작했다 [6, 7].

미 항공우주국(NASA: National Aeronautics and Space Administration)은 2000년도부터 우주선에 탑재되는 항공시스템의 전체적인 안전성을 보증하기 위해 각 기능을 제어 및 관리하는 데오스 커널(DEOS: Dynamic Enforcement Operating System)에 대한 정형검증을 수행하였다 [8, 9].

DEOS에 대한 연구는 다양한 기능을 수행하는 워커

Fig. 2. Pathfinder & Ariane 5 Failure

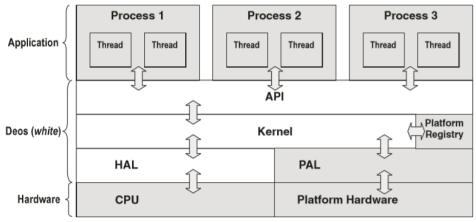

Fig. 3. DEOS Kernel Architecture

스레드가 CPU를 동일한 시간 동안 할당받아 태스크를 주기적으로 수행하는 것을 정형검증함으로써 시스템이 매 순간 안정적으로 동작할 수 있음을 보증하였다.

이와 같이 초기 정형검증 연구는 스케줄링 모듈의 정확성(Correctness)에 대한 연구가 중점적으로 진행되었지만, 점차 시스템이 복잡해지고 다양한 외부 인터페이스가 각 모듈에 추가되면서 스케줄링 이외에도 파일시스템 [10, 11], 하이퍼바이저 [12 - 14] 등 다양한 모듈에 대한 정형 검증하는 연구가 이루어지기 시작했다.

점차 시스템에 대한 정형검증 연구가 활발히 진행되면서 특정 커널 모듈뿐 만 아니라 전체 시스템에 대한 보안 수준을 보증하기 위한 연구가 수행되기 시작했다. 2009년 C언어로 작성된 L4 마이크로커널을 함수형 언어인 Haskell과 Isabelle/HOL을 통해 정형검증을 수행한 연구 결과가 발표되었다. 또한 시스템 콜 그래프에 따라 커널을 32 계층으로 분리하여 Coq로 정형검증을 수행한 연구도 발표되었다 [15 - 18].

하지만 기존의 연구는 대부분 커널이 명세대로 동작하고 있음을 검증하는 Correctness에 대한 연구에 편중되어 있으며, 보안을 고려한 기능을 추가 고려하여 검증한 연구 사례는 아직까지 보고되지 않고 있다.

III. 군사용 드론시스템 시스템 공통 구성요소 도출

미, 영 뿐만 아니라 NATO(North Atlantic Treaty Organization)는 운용 중인 군용 드론시스템을 Brooke-Holland가 제안한 기준에 따라 Table 1과 같이 분류한다 [19, 20]

Table 1. Classification for Military Drone

Category	Identifier	Weight
Class I(a)	Nano drone	Under 200g
Class I(b)	Micro drones	Between 200g and 2kg
Class I(c)	Mini drones	Between 2kg and 20kg
Class I(d)	Small drones	Between 20kg and 150kg
Class II	Tactical drones	Between 150kg and 600kg
Class III	MALE/HALE drones	Over 600kg

군용 드론에 탑재되는 센서 및 구성 하드웨어는 아래 Table 2와 같다. 단기 정찰 임무를 수행하기 위해 시스템 구성 모듈을 최소화한 Nano drone에 속하는 Black Hornet Nano 드론부터 대형 무인정찰기와 같이 장시간 동안 목표물 감시를 위해 운용되는 Tactical 또는 MALE/HALE drone의 공통적인 시스템 구성요소를 도출한다.

상기 군용 드론시스템 별로 탑재되는 시스템 구성요소의 공통점은 2가지이다. 첫째, 정찰 임무를 수행하기 위해 카메라, GPS 센서, 자이로/가속도 센서, 모터를 사용한다. 둘째 통신 모듈을 통해 수집된

Table 2. Comparison for Military Drone System

Category	Model	System sub-Component											
		Sensor						Communication		Motr		Fuel	
		C	O	G	A	T	P	RF	S	E	G	E	G
Class I(a)	Black Hornet Nano	O	X	O	O	X	X	O	X	O	X	O	X
Class I(b)	Perdex.	O	X			X	X	O	X	O	X	O	X
Class I(c)	T-Rotors	O	X	O	O	X	X	O	X	O	X	O	X
Class I(d)	RemoEye -002B	O	X	O	O	X	X	O	X	O	X	O	X
Class II	Heron	O	O	O	O	O	O	X	O	X	O	X	O
Class III	MQ-9 Reaper	O	O	O	O	X	X	X	O	X	O	X	O
Common Components		∨		∨	∨			△		△		△	

C: Camera, O: Optic, G: Gyro, A:Accelerometer, T: Temperature, P: Pressure, E: Electric
G: Gasoline, RF: Radio Frequency, S: Satellite, ∨: Common, △: Depends on System

정보를 전송하거나 원격으로 수동 제어될 수 있다. 이때 모터 및 통신 모듈은 시스템 규모에 따라 활용 요소의 종류가 상이할 수 있다.

최종적으로 도출된 군사용 드론시스템은 아래 Fig 4와 같은 구조를 가진다. 본 논문에서는 ChibiOS를 통해 도출된 공통 시스템 구성요소를 제어하는 군사용 드론시스템에 대한 보안성을 분석한다.

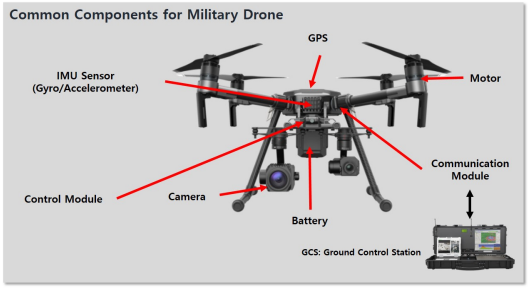

Fig. 4. Common components for Military Drone

IV. 보안위협모델링

분석하고자 하는 시스템 내 모든 보안위협을 체계적으로 도출하는 방법론으로 STRIDE 보안위협모델링을 사용한다. 보안위협모델링이란 체계적인 프로세스를 통해 분석하고자 하는 시스템의 전반적인 보안 위협을 모두 식별하는 방법이다 [21, 22]. 그중 일반 대중에게 상대적으로 잘 알려진 STRIDE 보안위협모델링은 Microsoft 사 빌 게이츠 회장의 메모로부터 제안된 점차 개선된 방법으로 6가지 보안 상 위협측면에서 시스템을 분석한다. 해당 보안위협은 다음과 같다.

- Spoofing(위장): 임의의 주체 또는 시스템 구성 요소가 인증을 정상적으로 획득한 것처럼 위장하는 위협
- Tampering(변조): 프로세스 내 또는 유·무선 통신을 통한 전송 중 데이터가 위·변조되는 위협
- Repudiation(부인): 임의의 주체가 특정 객체에 대해 쓰기/읽기/실행과 같은 오퍼레이션을 수행한 뒤 이를 부인하는 위협
- Information disclosure(정보 노출): 사용자 또는 임의의 프로세스의 중요 정보가 비인가된 접근에 의해 노출되는 위협
- Denial of Service(서비스 거부): 자원 소모나 독점과 같은 비정상적인 오퍼레이션에 의해 정상적인 동작이 실행되지 못하는 위협
- Elevation of Privilege(권한 상승): 낮은 권한을 가진 주체가 상대적으로 높은 권한이 요구되는 객체에 접근하여 특정 행위를 수행하는 위협

데이터 각각의 시스템 개발 단계에서 발생 가능한 위협을 모두 식별하고 이를 완화하기 위해 사용되며 크게 '기능 및 데이터 흐름 파악', '보안 위협 식별 및 공격트리', '완화 방안 제시' 3가지 단계로 구분할 수 있다.

4.1 기능 데이터 흐름 파악

본 장에서는 드론시스템에 대한 보안위협모델링을 적용한 분석 결과를 제시한다. 데이터 흐름도(DFD: Data Flow Diagram)는 아래 표 3에 있는 요소를 활용하여 작성자가 시스템 구성 정보에 접근할 수 있는 수준에 따라 더욱 상세하게 작성될 수 있다.

분석 대상인 드론시스템의 데이터 흐름은 상세 수준에 따라 Level 0부터 Level N까지 다양한 수준으로 구분되어 표현된다.

아래 Fig 5는 드론시스템 구동 후, 단일 커널 생성되는 커널 스레드와 사용자 스레드를 제어하는 데이터 흐름을 간략하게 표현한 level 1 수준의 데이터 흐름도이다.

탑재된 드론시스템에서 스레드는 3가지 상태

Table 3. Components of DFD

Element	Symbol	Description
Process	Process	System Code
Data Store	Data Store	Any Library that store data
Data Flow	→	Communication between process
Entity	External Entity	People or User
Trust Boundary		Required level of trust

(RUN/ READY/ NOT_EXIST)로 구분된다. 아래 Fig 9와 같이 사용자 스레드는 외부에서 발생되는 이벤트나 소프트웨어 인터럽트에 따라 커널 프로세스에 의해 CPU를 할당받거나 스케줄링 큐에서 재실행되는 것을 기다린다.

4.2 보안 위협 식별 및 공격트리

커널 상 데이터 흐름에서 발생 가능한 수 있는 보안 상 위협은 아래 Table 4와 같이 총 62개가 식별되었다.

식별된 위협 이외에도 CVE [23], CWE [24]와 같이 보안 상 취약점이 기록된 데이터베이스를 통해 커널 상 발생할 수 있는 모든 보안 상 결함 및 취약점 정보를 추가 수집한다. 수집 결과를 통해 아래 Fig 5와 같이 드론시스템에서 실제 발생할 수 있는 공격 경로를 다음과 같은 트리 형태로 작성하였다.

식별된 위협 이외에도 보안 상 취약점 정보가 기록된 데이터베이스나 최신 연구 논문 및 기술백서 등을 통해 커널 상 발생할 수 있는 모든 보안 상 결함 및 취약점 정보를 수집한 공격 라이브러리를

Table 4. Identified Threats

Category	ID	Type	Detail
E1	T1	S	Flight command can be sent to drones who are legitimate E1
	T2	I	System information (H/W configuration and thread information) about the drones under control can be exposed to the outside
E2	T3	S	Incorrect configuration could cause your app to run in kernel domain
	T4	D	Programs written by the programmer may degrade system Capability
P1	T6	I	An attacker can easily access the kernel (stack/heap) address offset by not using virtual memory
	T7	E	Programmers can create kernel threads in user area
P2	T8	S	The attacker can pretend to have obtained debugging privileges
P3	T9	E	A legitimate Interrupt Service Routine (ISR) inherited interrupt handler can be executed
P4	T10	S	The attacker can pretend to have obtained debugging privileges
	T11	I	System timer information can be exposed to outside
	.		
	.		
	.		
I1	T27	I	Thread execution information can be leaked by overflow error
	T28	E	User can gain debugging privilege by malicious app
I2	T29	D	The kernel may not be able to create threads due to memory deficit
	T30	E	Incorrect configurations can cause the thread to run with kernel privileges
I3	T31	S	Malicious apps may request legitimate app inheritance
	T32	E	When an interrupt occurs, ISR can be executed before the kernel service
I4	T33	S	Any thread can be preemptively executed by thread of user privileges
	T34	T	Scheduling queue data can be tampered, and thread execution order can change
	T35	D	An arbitrary thread may not be allocated a CPU due to a continuous occurrence of a timeout or delay signal by an attacker
	.		
	.		
	.		
D5	T61	T	Attacker upload malicious file with backdoor installed instead of normal system configuration file
D6	T62	T	Attacker upload malicious file instead of normal sensor file

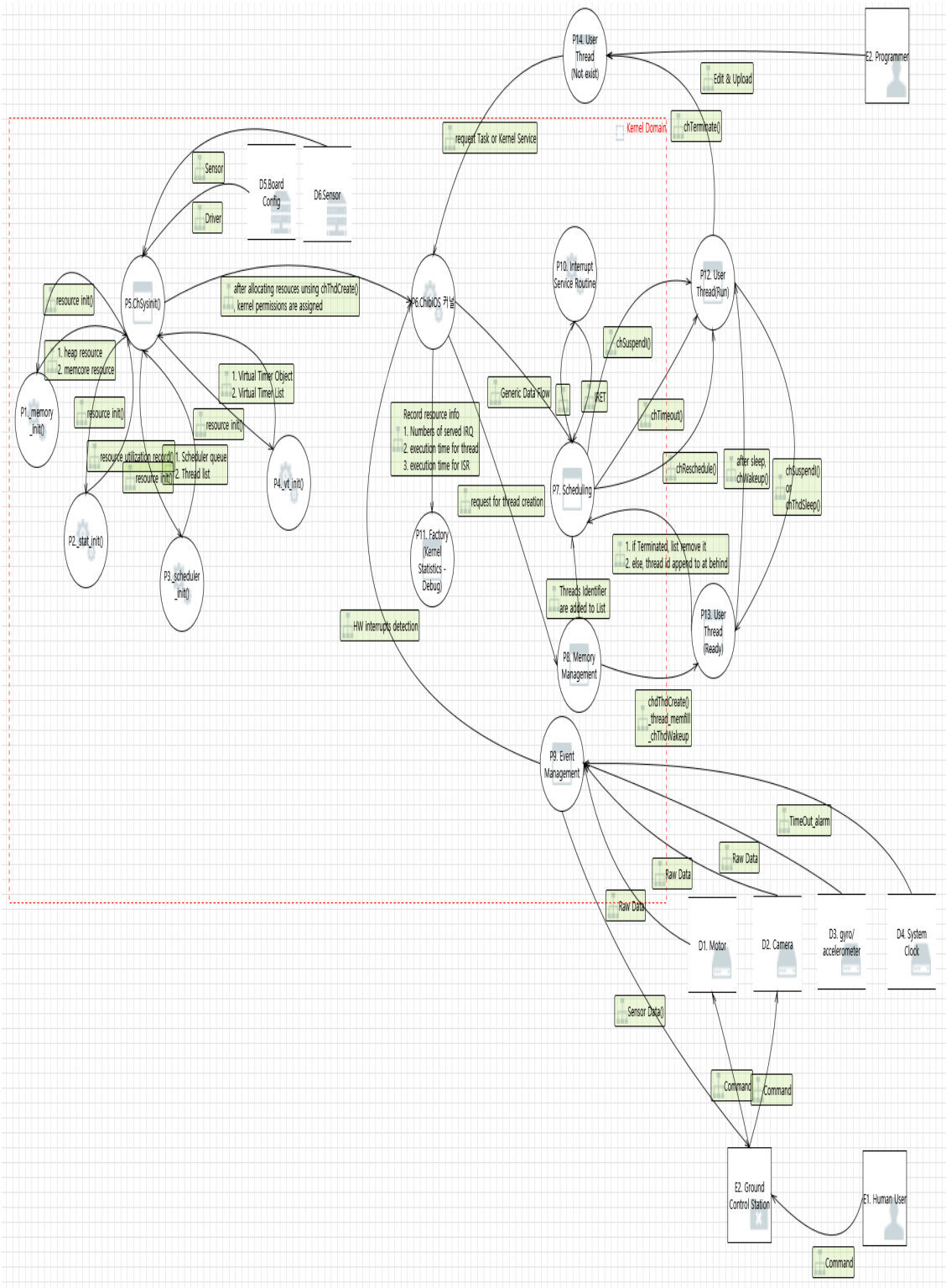

Fig. 5. Level 1 DFD

Table 5. Attack Library

Category	Author	Title	Reference
Public	MITRE	CVE (Common Vulnerabilities and Exposures)	[23]
	MITRE	CWE (Common Weakness Enumeration)	[24]
	MITRE	CAPEC (Common Attack Pattern Enumeration and Classification)	[25]
	OWASP	Embedded Application Security	[26]
Paper	Samland, Fred et al.	AR. Drone: security threat analysis and exemplary attack to track persons	[27]
	Rodday et al.	Exploring security vulnerabilities of unmanned aerial vehicles	[28]
	Shepard, Daniel P et al.	Drone hack: Spoofing attack demonstration on a civilian unmanned aerial vehicle	[29]
		⋮	
Tech Report	Rob Kidner	The security drones report 2017	[30]
	911 Security	AIRSPACE SECURITY INSIGHTS REPORT	[31]
	Sarah Ludwig	Drones: A Security Tool, Threat and Challenge	[32]
	US Dept of Transport	Unmanned Aircraft System (UAS) Service Demand 2015 - 2035	[33]

작성한다. 해당 결과는 다음과 같다.

4.3 완화 방안 제시

드론시스템에서 발생할 수 있는 사이버 공격은 위 Fig 6과 같이 총 10 가지이며 그에 대응하는 완화 방안은 아래 Table 6과 같다.

○ Acquire Kernel Privilege

- AP1: 개발자가 보안 상 취약한 함수를 사용하거나 사용자가 시스템 잘못된 시스템 설정 값을 적용함으로써 악용할 수 있는 공격 방법이다. 이를 완화하기 위해 개발자/시스템 사용자에 대한 추가적인 교육을 수행한다[34, 35].

- AP2, 3: 시스템 메모리 구조 상 취약점으로 인해 커널 실행 정보가 저장된 태스크 제어 블록(TCB: Task Control Block) 데이터가 공격자에 의해 위·변조되거나 노출되는 위협이다. 이를 완화하기 위해 기존 시스템은 MMU (Memory Management Unit) 또는 MPU (Memory Protection Unit)을 활용한 가상화 기법이나 컴파일러 측면에서의 스택/힙 오버플로우 공격을 통한 이상행위를 탐지하는 Stack Canary, DEP/NX 기법을 활용한다[36 - 39].

○ Denial of Service

- AP4, 5: 프로세스 생성 함수를 연속적으로 발생하거나, 시스템 또는 탑재된 프로그램 상 논리 오류로 인해 임의의 프로세스가 CPU 또는 메모리를 독점하는 보안위협이다. 이를 완화하기 위해 커널 측면에서 다양한 개선이 이루어지고 있다. 실시간 운영체제인 QNX와 같이 커널 내 타이머 제어모듈을 개선하거나 일반 운영체제인 리눅스와 같이 시스템 태스크 및 자원 제어 모듈을 개선하는 방법이 활용되었다 [40 - 43].
- AP 6, 7, 8, 9, 10: 해당 보안위협은 SYN, ICMP, UDP Flooding등과 같이 네트워크 자원을 소진시킴으로써 서비스 처리 속도를 저하시키는 보안위협이다. 이를 완화하기 위해

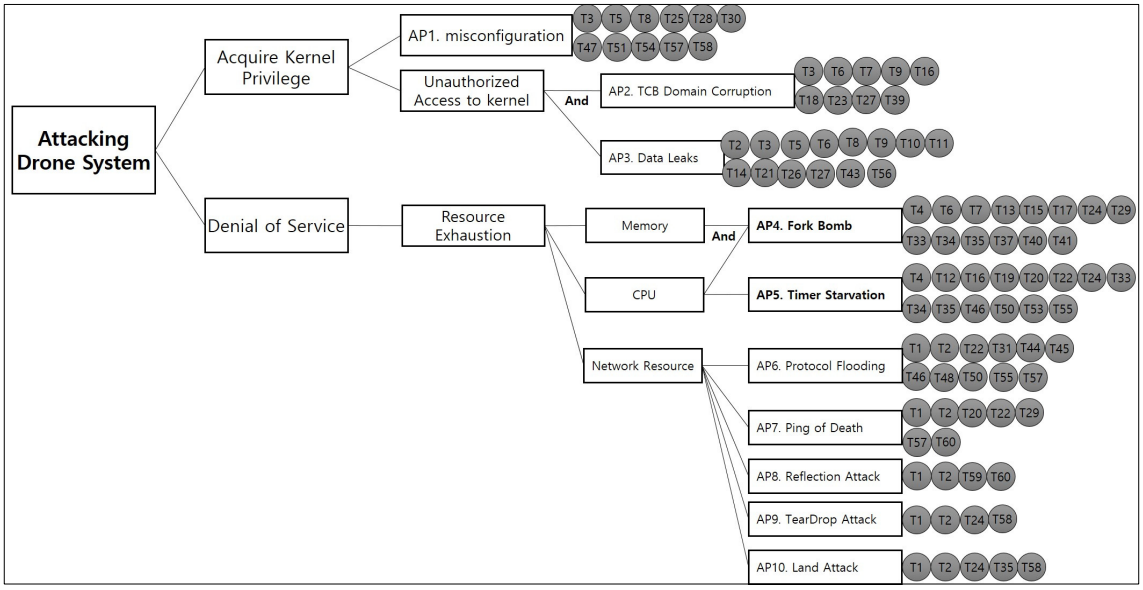

Fig. 6. Attack Tree for Military Drone System

아마존을 포함한 다양한 보안 백서에서는 IDS/IPS 영역에서 이상행위를 탐지하는 모듈을 사용하는 것을 제안하고 있다[44 - 46].

비정상적인 커널 권한을 획득할 수 있는 AP1, 2, 3과 같은 사이버공격은 공격 원인이 개발자 및 사용자 행위나 하드웨어 메모리 관리 방식 중 보안상 취약점을 통해 발생한다. 서비스 거부 공격을 유발하는 공격 중 AP4, 5의 경우, 커널 내 자원 관리 모듈 상 결함으로 인해 발생하고 AP6 ~ 10은 전송된 패킷 처리 방식 상 결함으로 인해 시스템 오류를 발생시킨다. 따라서 상기 제시된 각 공격에 대한 원인을 제거하기 위한 방식 중 커널 측면에서 완화할 수 있는 사이버공격은 AP4, 5에 해당된다.

일반적인 시스템은 해당 공격에 대응하기 위해 fork()와 같은 시스템 콜을 통해 단순히 생성

Table 6. Mitigation Method for Attacks

Category	Detailed Attack	Mitigation Method	
		Type	Details
Acquire kernel Privilege	AP1. Misconfiguration	Education	1. Training Developer and User
	AP2. Memory Corruption	Memeory Protection method	1. Virtual Memory 2. ASLR, Canary, DEP/NX etc
	AP3. Data Leaks		
Denial of Service	AP4. Fork Bomb	Kernel	1. Fix bugs in kernel config files (ex: /etc /security /limits.conf)
	AP5. Timer Starvation	Kernel	1. Fix bugs in Timer source codes
	AP6. Protocol Flooding	IDS/IPS	1. Detect Excessive packet rate 2. Adding Detail rules in IDS/IPS 3. Validate Timerout option
	AP7. Ping of Death	IDS/IPS	1. Restrict ICMP packet through ping command 2. Adding Detail rules in IDS/IPS
	AP8. Reflective DoS Attack	IDS/IPS	1. Add more network device to increase bandwidth
	AP9. Teardrop Attack	IDS/IPS	1. Check packet fragment offset 2. Adding Detail rules in IDS/IPS
	AP10. Land Attack	IDS/IPS	1. Check for source and destination IP 2. Adding Detail rules in IDS/IPS

가능한 스레드 수만 제한한다. 하지만 카메라, GPS, Wi-Fi 등 다양한 센서가 장착된 드론의 경우 단순히 생성되는 스레드 수를 제한한다면 주기적으로 발생하는 태스크를 정상 처리 및 종료할 수 없다. 따라서 본 논문에서는 실행 스레드 수에 따라 동적으로 CPU 실행 시간을 조절할 수 있는 스케줄링 방식을 제안하고 이에 대해 정형명세 및 검증을 수행한다.

V. 고신뢰 드론시스템 스케줄링 방식 제안

라운드로빈 스케줄 방식은 선점형 스케줄링의 일종으로 태스크 간 우선순위를 두지 않고, 동일한 시간 CPU를 할당하는 알고리즘이다. 라운드로빈 스케줄링 방식은 실행되는 태스크 실행에 대한 지연시간을 최소화하는 다른 실시간 스케줄링 방식과 다르게 모든 태스크가 동일한 우선순위를 가지고 동일한 횟수로 실행되는 fairness를 우선시한 방식이다.

따라서 라운드로빈 스케줄링 방식은 fork bomb나 Timer Starvation과 같이 현재 커널에서 실행되는 앱의 논리 오류로 인해 임의의 프로세스가 CPU를 독점하는 경우, 다른 태스크를 일정 기한 내 완료하지 못할 수 있다는 단점이 존재한다.

이를 보완하기 위해 본 논문에서 제안하는 시스템은 스레드 별 태스크 처리 횟수를 고려하여 각 태스크 CPU 할당 시간을 동적으로 조절하는 방식을 사용하고자 한다. 상기 방식은 태스크를 스레드 식별자(th_id)로 구분한다. 시스템은 해당 태스크를 수행하기 위해 스레드가 생성 시점부터 지금까지 얼마나 CPU를 할당받았는지 점검하고 이를 현재 레디 리스트에 저장된 스레드들과 그 값을 비교한다. 생성 후 처음 실행되는 스레드의 경우 초기 CPU 실행 시간 동안 태스크를 수행할 수 있도록 설정하고 실행이 반복되는 만큼 그 시간을 단계적으로 줄여나간다.CPU 오른쪽 Fig 7은 본 논문에서 제안하는 스케줄링 방식을 의사코드 형태로 나타낸 그림이다.

VI. Z를 이용한 알고리즘 정형 명세 및 검증

Z 표기법은 시스템의 상태를 표현하는 명세언어로 1977년 Oxford 대학 Jean-Raymond Abrial에 의해 개발되었다. 집합론, 일차논리에 근거하여 시스템의 상태와 각 상태 간 발생할 수 있는 시스템 동작을 스키마(Schema)로 명세하여 시스템에 대한 일관성, 완전성을 검증할 수 있다. Z 표기법은 오랜 기간 동안 강력한 표현력을 바탕으로 철도시스템, 스마트카드, Xen Hypervisor, 파일시스템 등 다양한 분야의 정보시스템을 정형 검증하는데 활용되었다[47 - 50].

Z 표기법은 스키마(Scheme)라는 기본 명세 구조를 가진다. 스키마는 시스템 상태를 정의하고 상태 별 제약 조건을 부여하는 상태 스키마와 상태 간 변환 시 수행되는 여러 연산을 나타낸 동작 스키마로 구분된다.

스케줄링은 시스템 클락에 의해 스레드 실행 시간과 상태(NotExist/ Ready/ Running) 간 전이가 발생하는 구조이기 때문에 상태 기반 언어인 Z언어를 통해 정형명세 및 검증을 진행하는 것이 적절하다. 따라서 본 논문에서는 서비스 거부 공격에 대한 완화 방법으로 제안하는 스케줄링 방식을 Z언어를 통해 정형명세하고 이를 Z언어 전용

```
Scheduling Module(Thread th_id)
th_id.state  <- Running                          // Thread State for executing is turned into running
pop Thread th_id from Ready Queue                // Thread for Executing are popped from Ready Queue

    for i<-0 to (Max_CPU Execution Time - th_id.count)
        Task t is executed                       // CPU is allocated for Thread whose state is running during allocated time

if (Task t is unfinished)                        // If thread is unfinished, Reschedule process are needed
    Task.count++                                 // The number of clock reset value's are counted
    Task.State <- Ready                          // During reschedule, thread state is returned into Ready
    Push(th_id)                                  // Push Thread id to Ready Queue Backward
    schedule(th_id.next)                         // Call next thread

else
    Task.state <-notexist                        // Else, terminate process are needed
    Task.count <-0                               // The number of clock reset value's are set to 0
    terminate(th_id)                             // Terminate thread
    schedule(th_id.next)                         // Call next thread
```

Fig. 7. Pseudo code for advanced scheduling process

정형검증 도구인 Z/EVES 2.3으로 검증한다.

앞서 제안한 스케줄링 방식에 대한 정형검증을 수행하기 위해 상태, 동작 스키마를 활용하여 아래와 같이 크게 3가지를 정형 명세한다.

- ○ 시스템 제약 조건: 스케줄러 상에서 허용되는 하위 태스크 간 상태 전이를 명세한다. 다음과 같이 정의되는 규칙들은 System Invariant로써 정형검증 수행 시, 각 시스템 상태 전이 중 상기 규칙이 위배되는지 점검하는데 활용된다.
- ○ 초기 상태: 시스템의 초기 상태를 명세한다.
- ○ 스케줄링 모듈: 스케줄링에 필요한 동작과 상태를 정형 명세하는 부분으로 선언부와 제약부로 구분된다. 선언부는 해당 상태나 동작을 명세하기 위해 필요한 하위 구성요소를 정의하는 부분이다. 제약부는 해당 앞서 선언부에서 정의한 하위 구성요소가 해당 상태를 만족하기 위한 조건을 정의하는 부분이다.

6.1 스케줄링 모듈 정형 명세

상위에서 기술한 개선된 스케줄링 방식은 CZT (Community Z Tool)를 이용하여 Z 표기법으로 정형명세하였다. CZT는 Z 표기법을 활용하기 위해 3가지 종류(Latex/UTF-8/UTF-16)의 인코딩 방식을 제공한다. 본 논문에서는 향후 정형검증에 사용되는 도구인 Z/EVES와 호환하기 위해 Latex 인코딩 방식으로 명세를 진행하였다.

아래 Fig 8은 명세한 스케줄링 방식에서 태스크가 가질 수 있는 상태들을 정의한 것이다. 태스크 상태는 태스크가 요청되지 않음을 표현하는 notexist, 현재 수행 대기 상태임을 표현하는 ready, CPU 할당 후 태스크가 수행됨을 표현하는 running 중 한가지로 표현될 수 있다. 또한 해당 상태 간 허용되는 전이 관계를 명시적으로 선언한 시스템 제약 조건이다.

```
\begin{zed}
  STATE ::= notexist | ready | running
\end{zed}
\begin{zed}
  StateTransition == (\{notexist\} \cross \{ready,running\})
  \cup(\{ready\} \cross \{ready,running\})
  \cup(\{running\} \cross \{ready,notexist\})
\end{zed}
```

Fig. 8. System Restriction

아래 Fig 9는 스케줄링 모듈 동작을 정형 명세한 것이다. 스케줄링 모듈은 선언부에서 Task 정보가 저장된 List 형태의 TaskData, 각 태스크의 상태 정보가 저장된 StateData, 시스템 클락 정보가 저장된 CountData, 태스크 수행 시간에 대한 정보가 저장된 RealTimeData 총 4가지 하위 구성요소를 정의된다. 또한 시스템 동작에 의해 상태가 변환될 때 4가지 Invariant가 위반되는지 정형 검증 시, 검증하게 된다.

아래 Fig 10은 스케줄링 모듈 초기 상태를 정형 명세하기 위해 하위 수준의 구성요소에 대한 초기화 동작을 명세한 것이다.

태스크 상태 변환이 발생하는 시스템 동작(Create/ Delete/ Run/ Suspend)에 대한

```
\begin{schema}{Scheduler}
  TaskData \\
  StateData \\
  CountData \\
  RealTimeData
\where
  taskset = TASK \setminus (state\inv \limg \{notexist\} \rimg) \\
  state \inv \limg \{running\} \rimg = \{runningtask\} \\
  FinishTime-StartTime \geq ExecutionTime \\
  ExecutionTime \geq Period
\end{schema}
```

Fig. 9. Formal Specification Scheduler

```
\begin{schema}{InitTaskData}
  TaskData'
\where
  taskset' = \emptyset \\
  runningtask' = \emptyset
\end{schema}

\begin{schema}{InitStateData}
  StateData'
\where
  state'=(\lambda x : TASK @ notexist)
\end{schema}

\begin{schema}{InitCountData}
  CountData'
\where
  count' = 0
\end{schema}

\begin{schema}{InitRealTimeData}
  ExecutionTime'
  StartTime'
  FinishTime'
  Period'
\where
  ExecutionTime = StartTime = FinishTime = 0 Period = 0
\end{schema}

\begin{schema}{InitTask}
  Scheduler'
\where
  InitTaskData \\
  InitStateData \\
  InitCountData \\
  InitRealTimeData
\end{schema}
```

Fig. 10. Formal Specification(Init)

정형 명세는 아래 Fig 11, 12와 같다.

```
\begin{schema}{TaskCreate}
  \Delta Scheduler \\
  target? : TASK
\where
  state(target?) = notexist \\
  taskset = taskset \cup \{target?\} \\
  StartTime = count \\
  runningtask' = runningtask \\
  state' = state \oplus \{(target? \mapsto ready)\}
\end{schema}
```

```
\begin{schema}{TaskDelete}
  \Delta Scheduler \\
  target?: TASK \\
  setTop!: TASK
\where
  state(target?) \in \{running\} \\
  state(setTop!) \notin \{notexist, running\} \\
  taskset' = taskset \setminus \{target?\} \\
  runningtask = setTop! \\
  state' = state \oplus \{(setTop!\mapsto running)
\end{schema}
```

Fig. 11. Formal Specification(Create/Delete)

```
\begin{schema}{TaskSuspend}
  \Delta Scheduler \\
  target?: Scheduler
\where
  state(target?) \in \{running\} \\
  FinishTime' = count \\
  state' = state \oplus \{(target? \mapsto suspended)\} \\
\end{schema}
```

```
\begin{schema}{TaskRun}
  \Delta Scheduler \\
  target? : Scheduler
\where
  state(target?) \in \{ready, suspended\} \\
  state' = state \oplus \{(target? \mapsto running)
  runningtask' = target?
  count' = count + CPUALLOCTIME
\end{schema}
```

Fig. 12. Formal Specification(Run/Create)

6.2 Z/EVES를 통한 정형검증

본 논문에서 제안하는 스케줄링 모듈의 정확성을 검증하기 위해 Z 표기법으로 명세한 시스템 모델에 대한 syntax checking, variable range checking 및 consistency type checking을 Z/EVES 2.3 도구를 통해 수행한다.

앞서 선언한 태스크 정보와 초기 시스템 상태 간 정형 검증한 결과는 아래 Fig 13과 같다.

```
\begin{theorem}{StateDataInit}
 \exists StateData' @ InitStateData
\end{theorem}

\begin{zproof}[StateDataInit]
 prove by reduce;
\end{zproof}
```

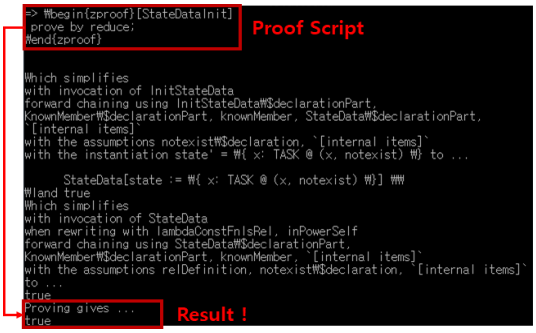

Fig. 13. Formal Verification

VII. 결 론

국가 차원의 대규모 사이버보안 위협이 증가하면서 외부 위협으로부터 드론시스템의 안전한 운용을 보증하기 위한 필요성이 대두되고 있다.

기존 드론시스템은 시스템 내부에서 발생할 수 있는 서비스 거부 공격에 대해 적절한 대응 방안을 가지고 있지 않다. 따라서 본 논문은 기존 서비스 거부 공격 중 Fork Bomb, Timer Starvation 공격을 완화하기 위한 새로운 스케줄링 방식을 제안하고 이를 Z 표기법으로 정형 명세 및 검증하였다.

향후에는 고신뢰 드론시스템을 제안하기 위해 서비스 거부 공격이외에도 드론시스템 커널 상 메모리 위·변조를 완화하기 위한 연구를 진행할 예정이다. 메모리 위·변조 위협을 완화할 수 있는 경량화된 메모리 보호 기법을 연구하고 이를 정형명세 및 검증하는 연구를 진행한다. 이를 통해 높은 수준의 보안성 및 안전성이 정형 명세 및 검증된 고신뢰 드론시스템에 대한 연구를 이어갈 예정이다.

References

[1] Adam Greenberg, "Facing Forward: Cyber Security in 2019 and Beyond",

FIREEYE, 2018
〔2〕 https://www.darpa.mil/program/high-assurance-cyber-military-systems
〔3〕 https://www.commoncriteriaportal.org/
〔4〕 Grimm, Tomás, Djones Lettnin, and Michael Hübner. "A survey on formal verification techniques for safety-critical systems-on-chip." Electronics 7.6 (2018)
〔5〕 Robert C. Armstrong, Ratish J. Punnoose, Matthew H. Wong, Jackson R. Mayo, "Survey of Existing Tools for Formal Verification" SANDIA REPORT
〔6〕 Kothari, Suresh, et al. "Modeling lessons from verifying large software systems for safety and security." Proceedings of the 2017 Winter Simulation Conference. IEEE Press, 2017.
〔7〕 http://www.cse.chalmers.se/~risat/Report_MarsPathFinder.pdf
〔8〕 Penix, John, et al. "Verification of time partitioning in the DEOS scheduler kernel." Proceedings of the 22nd international conference on Software engineering. ACM, 2000.
〔9〕 Penix, John, et al. "Verifying time partitioning in the DEOS scheduling kernel." Formal Methods in System Design 26.2 (2005): 103-135.
〔10〕 Kang, Eunsuk, and Daniel Jackson. "Formal modeling and analysis of a flash filesystem in Alloy." International Conference on Abstract State Machines, B and Z. Springer, Berlin, Heidelberg, 2008.
〔11〕 Bornholt, James, et al. "Specifying and checking file system crash-consistency models." ACMSIGARCH Computer Architecture News. Vol. 44. No. 2. ACM, 2016.
〔12〕 Barthe, Gilles, et al. "Formally verifying isolation and availability in an idealized model of virtualization." International Symposium on Formal Methods. Springer, Berlin, Heidelberg, 2011.
〔13〕 Freitas, Leo, and John McDermott. "Formal methods for security in the Xenon hypervisor." International journal on software tools for technology transfer 13.5 (2011): 463.
〔14〕 Leinenbach, Dirk, and Thomas Santen. "Verifying the Microsoft Hyper-V hypervisor with VCC." International Symposium on Formal Methods. Springer, Berlin, Heidelberg, 2009.
〔15〕 Klein, Gerwin, et al. "Comprehensive formal verification of an OS microkernel." ACM Transactions on Computer Systems (TOCS) 32.1 (2014)
〔16〕 Blackham, Bernard. Towards verified microkernels for real-time mixed-criticality systems. Diss. University of New South Wales, Sydney, Australia, 2013.
〔17〕 Gu, Ronghui, et al. "CertiKOS: An Extensible Architecture for Building Certified Concurrent {OS} Kernels." 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16). 2016.
〔18〕 KleinG, AndronickJ. "seL4: formal verification of a noperating system kernel." Communica tion of the Acm, 2010
〔19〕 Hassanalian, Mostafa, and Abdessattar Abdelkefi. "Classifications, applications, and design challenges of drones: A review." Progress in Aerospace Sciences 91 (2017): 99-131.
〔20〕 https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/432646/20150427-DCDC_JDN_3_10_Archived.pdf
〔21〕 Shostack, Adam. Threat modeling: Designing for security. John Wiley & Sons, 2014.
〔22〕 Shostack, Adam. "Experiences Threat Modeling at Microsoft." MODSEC@ MoDELS. 2008.
〔23〕 https://cve.mitre.org/index.html
〔24〕 https://cwe.mitre.org/index.html
〔25〕 https://capec.mitre.org/
〔26〕 https://www.owasp.org/

[27] Samland, Fred, et al. "AR. Drone: security threat analysis and exemplary attack to track persons." Intelligent Robots and Computer Vision XXIX: Algorithms and Techniques. Vol. 8301. International Society for Optics and Photonics, 2012.
[28] Rodday, Nils Miro, Ricardo de O. Schmidt, and Aiko Pras. "Exploring security vulnerabilities of unmanned aerial vehicles." NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2016.
[29] Shepard, Daniel P., Jahshan A. Bhatti, and Todd E. Humphreys. "Drone hack: Spoofing attack demonstration on a civilian unmanned aerial vehicle." (2012).
[30] Rob Kidner "The security drones report 2017", IFSEC GLOBAL 2017
[31] 911 Security. "AIRSPACE SECURITY INSIGHTS REPORT", Dedrone
[32] Sarah Ludwig. "Drones: A Security Tool, Threat and Challenge", SecurityMagazine
[33] U.S. Department of Transportation, "Unmanned Aircraft System (UAS) Service Demand 2015 - 2035 Literature Review & Projections of Future Usage". 2013
[34] Korea Internet & Security Agency (KISA), "Software Development Security Guide". 2017
[35] https://books.google.co.kr/books?id=Z9aNTafcb3IC&redir_esc=y
[36] Kuperman, Benjamin A., et al. "Detection and prevention of stack buffer overflow attacks." Communications of the ACM, 2005
[37] Robertson, William K., et al. "Run-time Detection of Heap-based Overflows." LISA. Vol. 3. 2003.
[38] Davi, Lucas, Ahmad-Reza Sadeghi, and Marcel Winandy. "ROPdefender: A detection tool to defend against return-oriented programming attacks." Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security. ACM, 2011.
[39] Wagle, Perry, and Crispin Cowan. "Stackguard: Simple stack smash protection for gcc." Proceedings of the GCC Developers Summit. 2003.
[40] https://www.cvedetails.com/cve/CVE-2002-1983/
[41] Nakagawa, Gaku, and Shuichi Oikawa. "Fork bomb attack mitigation by process resource quarantine." 2016 Fourth International Symposium on Computing and Networking (CANDAR). IEEE, 2016.
[42] Berlot, Michele, and Janche Sang. "Dealing with Process Overload Attacks in UNIX." Information Security Journal: A Global Perspective 17.1 (2008): 33-44.
[43] https://www.cyberciti.biz/tips/linux-limiting-user-process.html
[44] https://www.symantec.com/ko/kr/security-center/white-papers
[45] https://isis.kisa.or.kr/ebook/download_pdf/2018.pdf
[46] https://aws.amazon.com/ko/blogs/korea/aws-security-whitepapers/
[47] J. Woodcock, S. Stepney, D. Cooper, J. Clark, and 1. Jacob, "The certification of the Mondex electronic purse to ITSEC Level E6", Formal Aspects of Computing 20, 2008
[48] S. Stepney, and D. Cooper, "Formal Methods for Industrial Products", LNCS, Springer, ZB2000, York, Aug. 2000
[49] McDermott, and L. Freitas, "A Formal Security Policy for Xenon" ACM, Conference on Computer and Communications Security, Virginia, USA, 2008, pp. 43-52
[50] Praxis High Integrity Systems, "Tokeneer ID Station EAL5 Demonstrator: Summary Report", http://www.adacore.com/home/gnatpro/tokeneer, 2008

〈저자소개〉

곽 지 원 (Gil-dong Hong) 학생회원
2017년 2월: 중앙대학교 전자전기공학부 학사 졸업
2017년 3월~현재: 고려대학교 일반대학원 사이버국방학과 석사과정
〈관심분야〉 정보보호, 보안성 분석·평가, 커널 보안

강 수 영 (Soo-Young Kang) 학생회원
2006년 2월: 순천향대학교 컴퓨터공학부 공학사
2008년 2월: 순천향대학교 컴퓨터공학부 공학석사
2008년 5월~2010년 10월: 한국인터넷진흥원(KISA) 연구원
2010년 10월~2014년 10월: 안랩(Ahnlab) 주임연구원
2013년 3월~현재: 고려대학교 정보보호대학원 박사과정
〈관심분야〉 보안성 평가/인증, 위협 모델링, 소프트웨어 보안

김 승 주 (Seungjoo Kim) 종신회원
1994년~1999년: 성균관대학교 정보공학과(학사, 석사, 박사)
1998년~2004년: 한국인터넷진흥원(KISA) 팀장
2004년~2011년: 성균관대학교 정보통신공학부 부교수
2011년~현재: 고려대학교 사이버국방학과/정보보호대학원 정교수
2017년~현재: 고려대학교 사이버무기시험평가연구센터(CW-TEC) 부센터장
2004년~현재: 한국정보보호학회 이사
2007년: 국가정보원장 국가사이버안전업무 유공자 표창
2010년: 방송통신위원회 정보통신망 침해사고 민관합동조사단 위원
2011년~현재: (사)화이트해커연합 HARU 및 국제해킹대회 SECUINSIDE 설립자 및 이사
2012년: 선관위 디도스 특별검사팀 자문위원
2014년~2015년: 육군사관학교 초빙교수
2014년~2016년: 다음카카오 프라이버시 정책 자문위원회 위원
2015년~현재: 방위사업청 방산기술보호 자문관
2016년~2018년: 개인정보분쟁조정위원회 위원
2016년~현재: 산업통상자원부 전략물자기술 자문위원
2016년~현재: 한국카카오뱅크 정보보호부문 자문교수
2017년~현재: 국방보안연구소 정보보호분야 자문위원
2017년~현재: 여신금융협회 신용카드 단말기 시험 인증위원회 위원
〈관심분야〉 보안공학 및 SDL, 위협 리스크 모델링, 보안성 평가/인증, 암호학, Usable Security