TURUN YLIOPISTO
UNIVERSITY OF TURKU

ADAPTATION OF THE HUMAN NERVOUS SYSTEM FOR SELF-AWARE SECURE MOBILE AND IOT SYSTEMS

Nanda Kumar Thanigaivelan

TURUN YLIOPISTON JULKAISUJA – ANNALES UNIVERSITATIS TURKUENSIS
SARJA – SER. F OSA – TOM. 22 | TECHNICA – INFORMATICA | TURKU 2023

ADAPTATION OF THE HUMAN NERVOUS SYSTEM FOR SELF-AWARE SECURE MOBILE AND IOT SYSTEMS

Nanda Kumar Thanigaivelan

TURUN YLIOPISTON JULKAISUJA – ANNALES UNIVERSITATIS TURKUENSIS
SARJA – SER. F OSA – TOM. 22 | TECHNICA – INFORMATICA | TURKU 2023

University of Turku

Faculty of Technology
Department of Computing
Communication and Cyber Security Engineering
Doctoral Programme in Technology

Supervised by

Professor Seppo Virtanen
University of Turku

Adjunct Professor Ethiopia Nigussie
University of Turku

Professor Jouni Isoaho
University of Turku

Reviewed by

Professor Timo Hämäläinen
University of Jyväskylä
Finland

Professor Bruno Crispo
University of Trento
Italy

Opponent

Professor Juha Röning
University of Oulu
Finland

ISBN 978-951-29-9196-9 (PRINT)
ISBN 978-951-29-9197-6 (PDF)
ISSN 2736-9390 (PRINT)
ISSN 2736-9684 (ONLINE)
Painosalama, Turku, Finland, 2023

To my Dad, Mom, Gurus, and Goddess

UNIVERSITY OF TURKU
Faculty of Technology
Department of Computing
Communication and Cyber Security Engineering
THANIGAIVELAN, NANDA KUMAR: Adaptation of the human nervous system for self-aware secure mobile and IoT systems
Doctoral dissertation, 96 pp.
Doctoral Programme in Technology
March 2023

ABSTRACT

IT systems have been deployed across several domains, such as hospitals and industries, for the management of information and operations. These systems will soon be ubiquitous in every field due to the transition towards the Internet of Things (IoT). The IoT brings devices with sensory functions into IT systems through the process of internetworking. The sensory functions of IoT enable them to generate and process information automatically, either without human contribution or having the least human interaction possible aside from the information and operations management tasks. Security is crucial as it prevents system exploitation. Security has been employed after system implementation, and has rarely been considered as a part of the system. In this dissertation, a novel solution based on a biological approach is presented to embed security as an inalienable part of the system.

The proposed solution, in the form of a prototype of the system, is based on the functions of the human nervous system (HNS) in protecting its host from the impacts caused by external or internal changes. The contributions of this work are the derivation of a new system architecture from HNS functionalities and experiments that prove the implementation feasibility and efficiency of the proposed HNS-based architecture through prototype development and evaluation.

The first contribution of this work is the adaptation of human nervous system functions to propose a new architecture for IT systems security. The major organs and functions of the HNS are investigated and critical areas are identified for the adaptation process. Several individual system components with similar functions to the HNS are created and grouped to form individual subsystems. The relationship between these components is established in a similar way as in the HNS, resulting in a new system architecture that includes security as a core component. The adapted HNS-based system architecture is employed in two the experiments prove its implementation capability, enhancement of security, and overall system operations.

The second contribution is the implementation of the proposed HNS-based security solution in the IoT test-bed. A temperature-monitoring application with an intrusion detection system (IDS) based on the proposed HNS architecture is implemented as part of the test-bed experiment. Contiki OS is used for implementation, and the 6LoWPAN stack is modified during the development process. The application, together with the IDS, has a brain subsystem (BrSS), a spinal cord subsystem (SCSS), and other functions similar to the HNS whose names are changed. The HNS functions are shared between an edge router and resource-constrained devices

(RCDs) during implementation. The experiment is evaluated in both test-bed and simulation environments. Zolertia Z1 nodes are used to form a 6LoWPAN network, and an edge router is created by combining Pandaboard and Z1 node for a test-bed setup. Two networks with different numbers of sensor nodes are used as simulation environments in the Cooja simulator.

The third contribution of this dissertation is the implementation of the proposed HNS-based architecture in the mobile platform. In this phase, the Android operating system (OS) is selected for experimentation, and the proposed HNS-based architecture is specifically tailored for Android. A context-based dynamically reconfigurable access control system (CoDRA) is developed based on the principles of the refined HNS architecture. CoDRA is implemented through customization of Android OS and evaluated under real-time usage conditions in test-bed environments.

During the evaluation, the implemented prototype mimicked the nature of the HNS in securing the application under threat with negligible resource requirements and solved the problems in existing approaches by embedding security within the system. Furthermore, the results of the experiments highlighted the retention of HNS functions after refinement for different IT application areas, especially the IoT, due to its resource-constrained nature, and the implementable capability of our proposed HNS architecture.

KEYWORDS: Human nervous system, cross-layer design, access control system, intrusion detection system, context-aware policies, self-awareness, cybersecurity, internet of things, android

TURUN YLIOPISTO
Teknillinen tiedekunta
Tietotekniikan laitos
Tietoliikenne- ja kyberturvallisuusteknologia
THANIGAIVELAN, NANDA KUMAR: Adaptation of the human nervous system for self-aware secure mobile and IoT systems
Väitöskirja, 96 s.
Teknologian tohtoriohjelma
Maaliskuu 2023

TIIVISTELMÄ

IT-järjestelmiä hyödynnetään tiedon ja toimintojen hallinnassa useilla aloilla, kuten sairaaloissa ja teollisuudessa. Siirtyminen kohti esineiden Internetiä (Internet of Things, IoT) tuo tällaiset laitteet yhä kiinteämmäksi osaksi jokapäiväistä elämää. IT-järjestelmiin liitettyjen IoT-laitteiden sensoritoiminnot mahdollistavat tiedon automaattisen havainnoinnin ja käsittelyn osana suurempaa järjestelmää jopa täysin ilman ihmisen myötävaikutusta, poislukien mahdolliset ylläpito- ja hallintatoimenpiteet. Turvallisuus on ratkaisevan tärkeää IT-järjestelmien luvattoman käytön estämiseksi. Valitettavan usein järjestelmäsuunnittelussa turvallisuus ei ole osana ydinsuunnitteluprosessia, vaan otetaan huomioon vasta käyttöönoton jälkeen. Tässä väitöskirjassa esitellään uudenlainen biologiseen lähestymistapaan perustuva ratkaisu, jolla turvallisuus voidaan sisällyttää erottamattomaksi osaksi järjestelmää.

Ehdotettu prototyyppiratkaisu perustuu ihmisen hermoston toimintaan tilanteessa, jossa se suojelee isäntäänsä ulkoisten tai sisäisten muutosten vaikutuksilta. Tämän työn keskeiset tulokset ovat uuden järjestelmäarkkitehtuurin johtaminen ihmisen hermoston toimintaperiaatteesta sekä tällaisen järjestelmän toteutettavuuden ja tehokkuuden arviointi kokeellisen prototyypin kehittämisen ja toiminnan arvioinnin avulla.

Tämän väitöskirjan ensimmäinen kontribuutio on ihmisen hermoston toimintoihin perustuva IT-järjestelmäarkkitehtuuri. Tutkimuksessa arvioidaan ihmisen hermoston toimintaa ja tunnistetaan keskeiset toiminnot ja toiminnallisuudet, jotka mallinnetaan osaksi kehitettävää järjestelmää luomalla näitä vastaavat järjestelmäkomponentit. Nä-istä kootaan toiminnallisuudeltaan hermostoa vastaavat osajärjestelmät, joiden keskinäinen toiminta mallintaa ihmisen hermoston toimintaa. Näin luodaan arkkitehtuuri, jonka keskeisenä komponenttina on turvallisuus. Tämän pohjalta toteutetaan kaksi prototyyppijärjestelmää, joiden avulla arvioidaan arkkitehtuurin toteutuskelpoisuutta, turvallisuutta sekä toimintakykyä.

Toinen kontribuutio on esitetyn hermostopohjaisen turvallisuusratkaisun toteuttaminen IoT-testialustalla. Kehitettyyn arkkitehtuuriin perustuva ja tunkeutumisen estojärjestelmän (intrusion detection system, IDS) sisältävä lämpötilan seurantasovellus toteutetaan käyttäen Contiki OS -käytöjärjestelmää. 6LoWPAN protokollapinoa muokataan tarpeen mukaan kehitysprosessin aikana. IDS:n lisäksi sovellukseen kuuluu aivo-osajärjestelmä (Brain subsystem, BrSS), selkäydinosajärjestelmä (Spinal cord subsystem, SCSS), sekä muita hermoston kaltaisia toimintoja. Nämä toiminnot jaetaan reunareitittimen ja resurssirajoitteisten laitteiden kesken. Tuloksia arvioidaan sekä simulaatioiden että testialustan tulosten perusteella. Testialustaa varten 6LoW-

PAN verkon toteutukseen valittiin Zolertia Z1 ja reunareititin on toteutettu Pandaboardin ja Z1:n yhdistelmällä. Cooja-simulaattorissa käytettiin mallinnukseen ympäristönä kahta erillistä ja erikokoisuta sensoriverkkoa.

Kolmas tämän väitöskirjan kontribuutio on kehitetyn hermostopohjaisen arkkitehtuurin toteuttaminen mobiilialustassa. Toteutuksen alustaksi valitaan Android-käyttöjärjestelmä, ja kehitetty arkkitehtuuri räätälöidään Androidille. Tuloksena on kontekstipohjainen dynaamisesti uudelleen konfiguroitava pääsynvalvontajärjestelmä (context-based dynamically reconfigurable access control system, CoDRA). CoDRA toteutetaan mukauttamalla Androidin käyttöjärjestelmää ja toteutuksen toimivuutta arvioidaan reaaliaikaisissa käyttöolosuhteissa testialustaympäristöissä.

Toteutusta arvioitaessa havaittiin, että kehitetty prototyyppi jäljitteli ihmishermoston toimintaa kohdesovelluksen suojaamisessa, suoriutui tehtävästään vähäisillä resurssivaatimuksilla ja onnistui sisällyttämään turvallisuuden järjestelmän ydintoimintoihin. Tulokset osoittivat, että tämän tyyppinen järjestelmä on toteutettavissa sekä sen, että järjestelmän hermostonkaltainen toiminnallisuus säilyy siirryttäessä sovellusalueelta toiselle, erityisesti resursseiltaan rajoittuneissa IoT-järjestelmissä.

ASIASANAT: Ihmisen hermostojärjestelmä, ristikkäiskerroksinen suunnittelu, pääsynvalvontajärjestelmä, tunkeutumisen havaitsemisjärjestelmä, kontekstitietoiset säännöt, itsetietoisuus, kyberturvallisuus, esineiden Internet, Android

Acknowledgements

I wish to express my sincere gratitude to Professor Seppo Virtanen and Adjunct Professor Ethiopia Nigussie for being my supervisors, and Professor Jouni Isoaho for being my research mentor throughout my research career in the Department of Computing, University of Turku. I am always grateful for their guidance, expertise in the area, arrangement of facilities, freedom and flexibility to carry out the research.

I thank Professor Timo Hämäläinen from the University of Jyväskylä and Professor Bruno Crispo from the University of Trento for being my reviewers and for their constructive suggestions and recommendations for the improvement of the dissertation.

I am thankful to the University of Turku for supporting my research through employment, MATTI for providing grants to support my journeys to international conferences, and the University of Turku Graduate School (UTUGS) for funding my research. I thank Antti Hakala, D.Sc. (Tech.), for his assistance in research activities, and help in translating the abstract to Finnish. Furthermore, I would also thank my colleagues at the communication engineering and cyber security laboratory in the department for their support.

Finally, it is my greatest honor to convey my earnest gratitude to my dad, Mr. Thanigaivelan Mudaliyar and my mom, Mrs. Hemalatha Thanigaivelan, for their selfless support, countless sacrifices, and invaluable teachings, which have been helping me to stand and walk on my own feet. I also thank my elder brother, Mr. Sakthivelan Thanigaivelan for being my worst critic, an ardent well-wisher, and living proof of the embodiment of the establishment of a strong bond through years of sibling rivalry.

February 14th, 2023
Nanda Kumar Thanigaivelan

Table of Contents

Abbreviations

6LoWPAN	IPv6 over LoWPAN
ABAC	Attribute based Access Control
AC	Autonomic Computing
ACS	Access Control System
ADB	Android Debug Bridge
AES	Advanced Encryption Standard
AI	Artificial Intelligence
ANS	Autonomic Nervous System
AOSP	Android Open Source Project
API	Application Programming Interface
APK	Application Package Kit for Android
APKES	Adaptable Pairwise Key Establishment Scheme
ART	Android Runtime
AVB	Android Verified Boot
BLE	Bluetooth Low Energy
BrSS	Brain subsystem
CCA	Clear Channel Assessment
CCM	Counter with cipher block chaining message authentication code
CCS	Communication and Computing Device
CFL	Control Flow Integrity
CNS	Central Nervous System
CPU	Central Processing Unit
DAC	Discretionary Access Control
DAG	Directed Acyclic Graph
DAO	Destination Advertisement Object
DAO ACK	DAO Acknowledgement
DCI	Destination Context Identifier
DIO	DODAG Information Object
DIS	DODAG Information Solicitation
DODAG	Destination Oriented Directed Acyclic Graph
DoS	Denial of Service
DPO	Distress Propagation Object

DSCP	Differentiated Services Code Point
EBEAP	Easy Broadcast Encryption and Authentication Protocol
ECC	Elliptical Curve Cryptography
ECN	Explicit Congestion Notification
ED	Energy Detection
EID	Extension Header Identifier
EUI	Extended Unique Identifier
FBE	File-based Encryption
FFD	Full-function Device
FIB	Forwarding Information Base table
GMS	Google Mobile Services
GPU	Graphical Processing Unit
GTS	Guaranteed time slot
HAL	Hardware Abstraction Layer
HIDS	Host-based IDS
HIS	Human Immune System
HNS	Human Nervous System
ICC	Inter-Component Communication
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol version 6
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IPC	Inter-Process Communication
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IID	Interface Identifier
IoT	Internet of Things
IP	Internet Protocol
IPv6	Internet Protocol version 6
IPS	Intrusion Prevention System
ISS	Isolation Subsystem
JCA	Java Cryptography Architecture
JNI	Java Native Interface
L1	Layer 1 in the protocol stack, i.e., Physical Layer
L2	Layer 2 in the protocol stack, i.e., Link or MAC Layer
L3	Layer 3 in the protocol stack, i.e., Network Layer
LECIM	Low-Energy Critical Infrastructure Monitoring
LLN	Low-power and Lossy Network
LoWPAN	Low-power Wireless Personal Area Network
LQI	Link Quality Indication
LR-WPAN	Low-rate WPAN

MAC	Media Access Control
MCU	Microcontroller Unit
MGSS	Monitoring and Grading Subsystem
MMS	Multimedia Messaging Service
MOP	Mode of Operation
MP2P	Multipoint-to-point
MPDU	MAC Protocol Data Unit
ND	Neighbor Discovery
NIDS	Network-based IDS
NIO	Neighbor Information Object
NIST	National Institute of Standards and Technology
OEM	Original Equipment Manufacturer
OF	Objective Function
ONIO	Observed Neighbor Information Object
P2P	Point-to-point
P2MP	Point-to-multipoint
PAN	Personal Area Network
PDU	Protocol Data Unit
PDSU	PHY Service Data Unit
PHY	Physical Layer
PID	Process Identifier
PIO	Prefix Information Object
PNS	Peripheral Nervous System
TCP	Transmission Control Protocol
RBAC	Role-based Access Control
RCC	Rail Communication and Control
RCD	Resource-Constrained Device
RESS	Receptor-Effector subsystem
RFC	Request for comments
RFD	Reduced-function device
RIO	Routing Information Object
ROLL	Routing over LLN
RPL	IPv6 Routing Protocol for LLN
RSS	Reporting Subsystem
SAWS	Self-aware System
SCI	Source Context Identifier
SCSS	Spinal cord subsystem
SELinux	Security-Enhanced Linux
SNEP	Secure Network Encryption Protocol
SNS	Somatic Nervous System
SoC	System on Chip

SUN	Smart Utility Network
UDP	User Datagram Protocol
UI	User Interface
UID	User Identifier
uTESLA	Timed Efficient Streaming Loss-tolerant Authentication Protocol (the micro version)
WSN	Wireless Sensor Network
WPAN	Wireless Personal Area Network

List of Original Publications

This dissertation work is composed of the following original publications:

I N. K. Thanigaivelan, E. Nigussie, S. Virtanen and J. Isoaho. Towards Human Bio-Inspired Defence Mechanism for Cyber Security. IEEE Security and Privacy Workshops (SPW), pp. 276-280, 2018.

II N. K. Thanigaivelan, E. Nigussie, R. K. Kanth, S. Virtanen and J. Isoaho. Distributed internal anomaly detection system for Internet-of-Things. IEEE Annual Consumer Communications & Networking Conference (CCNC), pp. 319-320, 2016.

III N. K. Thanigaivelan, E. Nigussie, S. Virtanen and J. Isoaho. Hybrid Internal Anomaly Detection System for IoT: Reactive Nodes with Cross-Layer Operation. Security and Communication Networks, vol. 2018, Article ID 3672698, 15 pages, 2018.

IV N. K. Thanigaivelan, E. Nigussie, S. Virtanen and J. Isoaho. Conceptual Security System Design for Mobile Platforms Based on Human Nervous System. In Barolli L., Xhafa F., Hussain O. (eds) Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Advances in Intelligent Systems and Computing (AISC), vol. 994, pp. 437-446, 2019, Springer.

V N. K. Thanigaivelan, E. Nigussie, A. Hakkala, S. Virtanen and J. Isoaho. CoDRA: Context-based dynamically reconfigurable access control system for android. Journal of Network and Computer Applications (JNCA), vol. 101, pp. 1-17, 2018.

The original publications have been reproduced with the permission of the copyright holders.

1 Introduction

Since the early ages, information exchange has been an essential human activity. This exchange is performed through communication using various mediums or direct engagement with other parties. The medium of communication has greatly improved through the evolution of human intelligence and has progressed over time, especially over the past 150 years, progressing from the use of animals to copper wires, and eventually, to wireless forms of communication. These transitions heavily influenced the computing systems used in communication, which were initially meant for military purposes and later introduced for civilian purposes.

However, civilian systems are not designed as secure as military systems. The usage of civilian systems went far beyond anyone's imagination and have become a huge monetary income source for governments and involved organizations. Even the slightest degree of enhancement in users' comfortability has an enormous impact on revenue generation for all parties involved. Thus, users' comfortability with communication and computing system (CCS) becomes directly proportional to higher revenue generation for governments and business entities. Communication systems have evolved from fixed telephone connections backed by centralized telephone exchanges to small handheld mobile devices that use satellites and towers for signal transfer. Similarly, computing systems evolved from large desktops to portable laptops. Throughout these stages of enhancement, the power of the computing systems has been vastly enriched through continuous research and development. The fusion of communication and computation devices resulted in a new form of devices that can perform both functions simultaneously. In some cases, these hybrid personal mobile devices have more processing power than normal desktops or laptops. The evolution of CCS is illustrated in Figure 1.

These enhancements attract a wide range of audiences, from the general public to governmental and business organizations, who adopt and utilize these devices in their day-to-day activities. Unfortunately, this adaptation has presented lucrative opportunities for groups with malicious intentions to carry out espionage, abuse, theft, and the usage of information for their personal benefit. Initially, these forms of espionage were limited to governmental institutions who were preoccupied with discovering the secrets and new developments of the other country's institutions. Incidents of espionages such as *the Thing* [1] employed resonant cavity microphones [2], while other incidents used easy chair mark I and II [3], telephone bugging devices [4] for espionage. A discussion of the implications of governmental espionage is beyond the scope of this thesis.

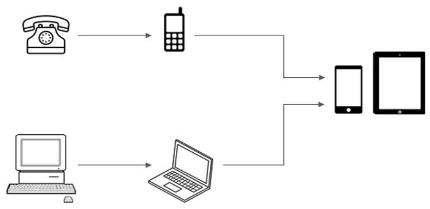


Figure 1. Evolution of communication and computing systems

On the civilian side, espionage has involved the theft of sensitive information which may result in direct or indirect monetary loss, breach of personal privacy, and, sometimes, loss of life. The primary motivation behind these activities is that the civilians pay for services rendered by business organizations through sending messages, making calls and surfing the internet using mobile devices. The organizations engaging in these activities are called telecom operators, and the civilians accessing these services through mobile devices are called mobile users or subscribers. The process of purchasing services from telecom operators is commonly referred to as a mobile subscription. The implications of espionage have been increased because the driving force of the enhancements is purely based on a swift market reach to reap enormous monetary gains, and in most cases, security has never been considered during the design phase of those systems. Hence, security became an afterthought, in a sense, patching over cracks in the foundation rather than reconstructing the building itself.

Furthermore, CCS has evolved to have sensors similar to, and in certain cases higher than human sensory perception. These sensors can sense multiple stimuli from the surrounding environment, including light, heat, surface vibration, quantity and chemical compositions in the air. They are deployed across several fields (e.g., hospitals, homes, offices and industries) for the automation and easy accomplishment of tasks without, or having the least degree of, human intervention. The size of these CCS devices is relatively small, and they communicate through a wireless medium; thus, their communication range is limited. These devices can recognize each other and operate together to fulfill the objectives of the application area. They can interconnect with each other to form networks and can be accessed from external IP networks, as each CCS can carry a unique address. This phenomenon is collectively referred to as the Internet of Things (IoT). Left unattended after implementation, these CCSs may become subject to physical tampering, converted into malicious CCSs, and introduced into the network to carry out attacks to disrupt operations. Since their mode of communication is wireless, their entire information exchanges are prone to eavesdropping and other forms of threat. In these cases, security mechanisms have used a tool to limit operational disruptions. Thus, security became an afterthought, even in handling these special kinds of CCS devices.

Through this research, a novel biological system architecture inspired by the function of the human nervous system (HNS) to address the problems in the existing approaches and embed security as one of the core system components. An experimental prototype will be built based on the newly proposed architecture and evaluated under two different IT domains as experiments to prove the enhancement in security, overall system operations and feasibility for implementation across several domains with different requirements and resource availabilities.

1.1 Motivation and objectives

The aim is to devise a methodology for implementation of built-in security that is adaptable to resource and operational requirements of the targeted systems. The primary motivations behind this work are to include security as a part of the system design process and to enhance the overall security of system operations through the employment of self-awareness. An increase in the security level of IT systems leads not only to stronger security but also to an increase in resources required for operations. Thus, security solutions should balance security and resource usage. In addition, they must be scalable and represent state of the art in order to work with future technologies. The objectives of this dissertation are as follows:

- To propose a design method for self-aware and adaptive built-in security mechanism.
- To develop a system architecture for self-aware and adaptive built-in security mechanism.
- To design, implement and evaluate the developed architecture in IoT and mobile test-bed systems as experiments.

1.2 Research Questions

Four research questions have been formulated to address the defined objectives:

- RQ1: How should one approach the introduction of self-awareness as part of system security?

Self-awareness requires an understanding of the operations within the system and its (surrounding) operational environment. Usage of artificial intelligence (AI) is one of the best approaches towards the realization of self-awareness. However, the employment of several AI agents to independently handle internal and external operations leads to problems (e.g., coordination). Through investigating this question, we aim to discover the common base for more than one AI by studying the human nervous system. These mechanisms are suitable for self-awareness as they tacitly adapt to any changes in their environment.

- RQ2: How does one build a new system or refine an existing system architecture that has security ingrained as a component through self-awareness?

The system architecture plays a significant role in the development of the system. It explains the organization of different layers, the presence of components in each layer and their interactions with each other. By answering this question, we aim to learn the changes that need to be carried out in order to include the security components, including what those changes are, how to execute them, what methods are used to execute them, the impacts of the changes on overall operations, and in which environments those changes are applied.

- RQ3: How does one generalize the proposed architecture so that it can be used in different application areas of IT (e.g., mobile and IoT)?

System development depends on architecture for the orderly execution and completion of its intended objectives. It is possible to secure the operations of different application areas of IT by following different architectures for each application area during implementation. In the end, they must work together to fulfill the objectives, otherwise, weak links might be established and possibly exploited. By exploring this question, we consider the possibilities of retaining the components and functionalities of the proposed architecture during the refinement process for different application areas.

- RQ4: How does one evaluate the threat mitigation and operational performance?

Several architectures were proposed, but most of them were not considered for real-life usage due to the complexity of the implementation process. Evaluation is necessary to determine the fulfillment of the objectives, and the proposed architecture must be implemented for the evaluation process. Through answering this question, we aim to discover what approach should be followed and how that approach can be justified with scientific means for the evaluation process. Thus, by combining the answers to RQ3 and RQ4, we develop a prototype as a proof-of-concept for the analysis of feasibility in practice and for evaluation to prove the threat mitigation and operational performance.

1.3 Research Methodology

Research methodology is a process of resolving a research problem by systematically using various approaches. The design science [5] approach is primarily followed in our work as it encourages the development of an artifact, which can be a prototype with limited or full capability, or a defined model, based on the definition of the problem. The developed artifact will undergo an evaluation process to prove that it solves the problem to a certain degree. We use the terms experimentation and proof-of-concept interchangeably to refer to this approach in our work. Our approaches to the research are as follows:

1. Study of the human nervous system and deriving a system architecture reflecting HNS functions: The entire scope of the HNS was studied thoroughly. Its features, such as spawning nerve networks, signal-gathering, decision-making, its controlling and protecting mechanisms and its relationship with other systems of the human body were examined exhaustively to answer questions such as a) What is unique about the HNS? b) Is decision-making centralized? c) Is protection active or passive in nature? After drawing conclusions to the questions above, a new system architecture comprised of components similar to the HNS, including its functionalities, was created. The new architecture was created such that it will retain HNS capabilities, even after severe refinement, to make it suitable for different experiments.

2. Experimentation of the proposed HNS architecture in IoT security and evaluation: The proposed HNS architecture will be analyzed under the IoT context to enhance its overall operations having security as part of the IoT system. The feasibility analysis was carried out to determine HNS architecture requirements and performances against the resource availability in IoT devices. This step is necessary, as IoT devices offer limited computational power and are expected to function for a prolonged period without human attention. Due to the resource-constrained nature of IoT devices, the proposed HNS architecture was refined concerning the IoT.

Evaluation of proof-of-concept in terms of threat-mitigation, performance and resource requirements was carried out. Three threat cases: packet flooding, selective forwarding and clone attacks, were used to assess the threat mitigation capabilities of our proposed architecture. The performance assessment utilized the factors of efficiency in spawning and latching nerve points, exchange of high- and low-speed signal and additional overhead in overall operations. A setup was established by incorporating sensors, an edge router and a server for the proof-of-concept test-bed. The edge router was constructed by combining PandaBoard [8] and Zolertia Z1 mote [9]. PandaBoard is based on Texas Instruments (TI) OMAP architecture. It is a single-board computer and a low-power development platform that employs the OMAP4430 system on chip (SoC). The SoC is comprised of a dual-core ARM Cortex-A9 MPCore 1 GHz CPU, a PowerVR SGX540 304 MHz GPU, a multimedia hardware accelerator and a 1 GB DDR2 SDRAM. The PandaBoard also comes with several different interfaces such as Wi-Fi, ethernet, Bluetooth, USB ports, HDMI and

DVI ports. Zolertia Z1 mote is used as sensor devices for constructing IoT networks. Z1 mote is equipped with a 16MHz ultra-low-power microcontroller unit (MCU), 92KB of flash memory, 8KB of RAM and an IEEE 802.15.4-compliant 2.4GHz CC2420 radio transceiver for communication purposes. Z1 mote was selected for this work to prove the operable capability of our refined HNS-based IoT architecture under low resource requirements. Contiki OS [10] is used for the implementation of the proposed HNS-IoT architecture and three threat cases for evaluation. Contiki is an open-source operating system (OS) for low-cost, low-powered devices that uses different communication stacks for connectivity purposes. A network simulator tool for Contiki OS called Cooja [11] and the host machine used as an edge router as well as a server was used for the simulation environment. Zolertia Z1 mote was emulated into the Cooja for the construction of IoT networks within the simulation environment.

3. Experimentation of the proposed HNS architecture in mobile security and evaluation: The proposed HNS architecture was studied in a mobile device environment. This study was necessary to understand the challenges and possibilities of refinement to prove the implementation capability and to evaluate the performance in terms of threat mitigation, interoperability, usability, resource requirements, and scalability of our proposed HNS-based architecture. Android [6] OS was chosen as a mobile platform for this study based on factors such as openly available source code and ease of deployment in the devices after customization. Android is an open-source mobile OS developed based on Linux kernel [7] that primarily targets touch-enabled user interface (UI) devices (e.g., smartphones, TVs, and tablets). Android can use multiple radio transceivers to handle communication such as cellular radio, Wi-Fi, and BLE. It also allows users to use applications for numerous purposes like calls, messages, and access to the internet. Android comes with built-in security mechanisms to prevent exploitation and abuse of its resources.

In the platform-level experiment, Android was personalized by ingraining refined HNS architecture. Thus, Android acquired organs and functionalities similar to the HNS. An HNS-based IDS was embedded into Android to establish links to the resources to trace the origin of resource-request calls and exert control over them. These links were exposed as reconfigurable policies. IDS was also made to work together with Android security mechanisms such as permission model and process isolation. A control panel is also implemented to provide an interface for configuration, decisions made and enforced, and recorded activities. For performance evaluation, the factors, effectiveness of observation and control links, employment of high and low-speed signals, and additional overhead in overall operations were utilized. The test-bed environment set was established using a Nexus 9 tablet, and Nexus 5 mobile phones were used for the proof-of-concept and evaluation.

1.4 Research contributions

The primary objective of this work is to enhance the security of computing devices by ingraining security inspired by the human nervous system (HNS) into their design. Therefore, contributions should include an adaptation of the human nervous systems and use a exploration study as a proof-of-concept to determine the usability and implementation capability of the adaptation. The contributions of this work are given in the original publications section and a brief overview of the list of contributions are presented below:

1. Adaptation of human nervous system functions and preparation for experimentation: The HNS plays a crucial role in the overall functioning of the human body. It acts as observers of events and carrier of messages by transmitting signals from different parts of the body to the brain via the spinal cord. The HNS also works as an enforcer by executing the decisions made by the brain and dorsal horn (spinal cord). The HNS comprises a huge network of nerves connecting tissues with the brain and spinal cord. The brain serves as a learning engine, decision maker, storage, and enforcer, while the spinal cord is a hub for nerve networks and decision-maker during critical circumstances.

The first contribution of this dissertation is the emulation of HNS operations into the system and security context by deriving an architecture based on the HNS organs of the brain, spinal cord, nerve networks, and communication signals. The aim of deriving an architecture are: i) to identify the placement possibilities of HNS organs and their functions in the system, ii) to study the advantages and shortcomings of the architecture including challenges during the utilization of the same in the real-time application domains. We also carried out the experiments using the proposed HNS-based architecture to analyze the challenges and possibilities in refinement during proof-of-concept development. Instead of building a system from an initial stage, we decided to customize the existing platforms for our work, and selected two popular platforms from two different fields of IT: a) Internet of Things and b) mobile devices. Availability of the source code, vast application areas, and popularity are the reason behind the selection of these two platforms.

2. Enhancement of security in IoT using proposed HNS architecture: The IoT describes a framework for interconnecting different physical objects over a communication medium for data exchange to implement several applications, for example, smart homes, smart cities, and smart transport systems. It is famous for operating under different conditions and for diverse purposes using various communication technologies. The IoT consists of resource-constrained devices (RCDs), low power networks, and actuators. RCDs are capable of using different communication technologies to exchange control messages to form a network topology and then observing and reporting their findings over a certain period. These RCDs communicate with each other and external networks over a wireless medium, which makes them vulnerable to multiple threats. The proposed HNS-based modified IoT platform is specifically tailored for RCDs and edge devices. Spinal cord functions operate from RCDs whereas brain functions execute in the edge device. With this modified IoT platform, RCDs and edge devices will acquire the ability to spawn and latch nerve points as control links over events occurring in the communication stack and processes. Through this process, they can:

(a) understand their own operations
(b) observe and learn nearby RCDs behavior
(c) employ high and low-speed signals for communication based on observed event sensitivity
(d) make and enforce decisions at the RCDs and edge device levels simultaneously
(e) seamlessly cooperate with other security mechanisms

We prove that the proposed architecture guarantees security against relevant attacks at both the individual and collective level. In addition, the results of the computation performance evaluation show that HNS functions introduced less overhead in terms of communication and behavioral pattern establishment. The evaluation also proved that the architecture requires low memory for its operation compared to the existing similar work.

3. Enhancement of mobile device operations using the proposed HNS architecture through platform customization: Mobile devices, especially smartphones, are extremely popular due to their ease of use. These devices are powered by several OS platforms, with Android and iOS being among the most popular. These devices also support communication with other networks using cellular radio, Wi-Fi, Bluetooth (BLE) and Near-field communication (NFC). Since these devices use monetary means in the form of subscription fees to use different services and handle a relatively large amount of personal information, they attracted threats with the aim of obtaining financial benefits or gaining personal information. Our proposed HNS-based architecture was refined to suit the mobile platform. HNS organs were placed within the architecture and armed to intercept communication between the architectural layers. This resulted in the customization of mobile platform architecture. We chose Android OS for this work and customized it for our proof-of-concept study. We also used Nexus 9 and Nexus 5 as evaluation devices. With our HNS architecture, the customized Android is capable of the following:

(a) spawning and latching nerve points at every request and function call originating from mobile applications

(b) employing low- and high-speed signals to communicate between the nerve points and HNS organs

(c) understand the nature of request and function calls through the usage signals from the nerve points

(d) enforcing decisions through nerve points

(e) cooperating with the Android's existing security mechanisms

Through this proof-of-concept study, we proved that the proposed architecture is implementable. During the evaluation, the above capabilities were fulfilled in the evaluation devices. Further evaluations were also carried out to measure the computational performance and memory requirements of the devices. The evaluation results proved that the overhead introductions during computation performance were meagre and memory requirements were very low compared to the other security mechanisms, including in-built security.

1.5 Thesis Organization

This dissertation is comprised of two parts. The first gives a detailed overview of the research in eight chapters, while the second part provides the original publications. The eight chapters of the first part are structured as follows:

- Chapter 1 presents the motivation and objectives of the research, describes the research questions and methodology, and presents an overview of research contributions.

- Chapter 2 introduces security definitions from two different standpoints and presents the definition of security that has been used in this dissertation. It also presents an overview of the security mechanisms used in information systems security and self-awareness.

- Chapter 3 describes the application areas (i.e., IoT and mobile devices) used in the experiments. The architecture, technologies, and standards for IoT and mobile devices are discussed in detail, along with their existing security implementations and proposed security approaches from the research community.

- Chapter 4 introduces natural and biological inspirations for science and technology, especially swarm intelligence and the human immune system. The architecture, organization, and functionalities of our inspiration, the human nervous system, are discussed comprehensively for the easy comprehension of our approach.

- Chapter 5 discusses the unique influential features of the HNS that can greatly enhance IT systems operations and security.

- Chapter 6 describes the contributions of this thesis in detail.
- Chapter 7 presents a summary of the original publications.
- Chapter 8 provides the conclusion and future research directions.

2 Security and Self-awareness

This chapter provides security definitions, a brief overview of self-awareness and the available security mechanisms of IT systems.

2.1 Security definitions and objectives

IT systems have grown rapidly, and their usage brings enormous benefits to every field (e.g., banking, hospital, and education) in terms of manageability, maintainability, comfortableness, and ease of task execution. They also generate, store, and maintain a wide variety of data. Any improper functioning of an IT system may result in system failure or loss of data, which in turn causes serious harm to the entire field or even loss of lives. Hence, security is a critical factor for the proper functioning of IT systems.

"Security is a system attribute that reflects the ability of the system to protect itself from external attacks that may be accidental or deliberate" [12]

"[Security] is protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide integrity, confidentiality and availability" [13]

The above two definitions clearly reflect the understanding of security from a software engineering and governmental standpoint. The former sees security as an element of the system itself whereas the latter treats security as a tool used to protect the system, but with a clear understanding of the objectives. Treating security as an element means that it should be developed alongside the development of the system. Combined with a clear definition of security objectives, this will prevent the need for additional tools. Henceforth, for this dissertation, security is defined as below, to reflect the above statement:

"Security should be an inalienable property of the system that protects the system's own functional resources, internal operations, and external communications from being abused. System security should co-exist and cooperate with external security means, if necessary"

Security objectives define the implementation of the required protection mechanisms based on the sensitivity of the information being handled by the IT systems.

Security objectives facilitate the security of the system functions by restricting attacks through the exploitation of vulnerabilities. Some of the commonly used security objectives are:

(a) Confidentiality: ensure that information is accessible only to authorized entities, processes or users.

(b) Integrity: guarantee that information has not been modified, destroyed or lost through accidental or unauthorized manner.

(c) Authentication: confirm the claim of having certain attributes by the system entity or resource.

(d) Authorization: approval to execute actions or access a system resource.

(e) Availability: make certain that system resources are available even during critical times.

(f) Access control: a property that prevents the unauthorized access or use of system resources.

As mentioned, defining a security objective is directly proportionate to the required security mechanism(s) and hence, it can be easily assumed that more security objective results in the achievement of additional security. Unfortunately, the application area of IT systems is vast and it plays a decisive role in the definition of security objectives. The primary reason is, not all IT systems are capable of accommodating multiple security mechanisms due to the availability of resources. Some application areas require data protection for a shorter period as the information will lose its importance. For example, in the IoT application area, the RCDs have limited resources and are expected to operate for a longer period unattended. Under these cases, the definition of security objectives should be made cautiously as it does not tax heavily on their resources and should guarantee security. For instance, the definition of access control property, for RCDs, in the form of a discretionary access control or mandatory access control for authorization won't create a secure environment for authorized access, rather it will increase operating costs and decrease the implementation feasibility and system performance. As an alternative, cryptography can be used to perform authorization together with confidentiality, integrity and authentication. Furthermore, security mechanisms should be simplified in order to use in IT application area, e.g., IoT, to match their resource availability and security requirements.

2.2 Security mechanisms overview

Security, as a tool or mechanism, exists in several forms, such as cryptography, firewall, anti-virus, intrusion detection system (IDS), and access control. These mechanisms are discussed briefly in the following subsections.

2.2.1 Cryptography

Cryptography is the oldest IT security mechanism used to conceal information from being exposed to adversaries. Encryption refers to the process of converting readable plain text into cipher text, whereas reconstruction of the original state from ciphertext is called decryption. Various cipher algorithms such as AES [117], ECC [118], RC5 [119] and RSA [223] are available for this purpose.

The cryptography algorithms are broadly classified into two categories based on their operations: a) block cipher divides plain text into multiple segments of fixed size (e.g., 128 bits) for ciphertext generation and b) stream cipher divides the plain text into successive stream elements and performs encryption of n-th input element using the same keystream element [190]. The cryptographic key is an important part of the encryption and decryption process. It further divides encryption into two general categories: symmetric and asymmetric. In symmetric encryption, the same key is used for both ciphering and deciphering processes, while asymmetric encryption employs two different keys [190]. The private and public key pairs are generated for asymmetric encryption. Public keys are published openly and can be accessed by anyone who wishes to send the encrypted message. The receiver will use their private key to reconstruct the original message.

2.2.2 Firewall and anti-virus

A firewall is an internetwork gateway mechanism used to create an external barrier around the network to thwart adversaries and malicious content from reaching the network. In addition, firewalls should permit authorized traffic from an internal network to an external network and vice versa [190]. Firewalls employ packet contents and attributes to perform their operations. For instance, a firewall can use packet header contents such as protocol types, port numbers, source address, destination address, and packet body contents for deep packet inspection process. A firewall is available as software (e.g., OPNsense [15], pfSense [14], Netfilter [16], and Cisco ASA [17]) and as hardware (e.g., Cisco Firepower [18] and Fortinet NGFW [19]). Generally, an IDS or intrusion prevention system (IPS) is also used, along with a firewall to deduce the suspicious behaviors from the network traffic.

Anti-virus is software-based security used to protect against known malicious programs, such as malware and ransomware. Anti-virus software should be installed in the target device that requires protection. It will automatically scan the objects (e.g., files) which are being accessed for malicious programs on regular intervals. Generally, anti-virus software uses signatures to execute its detection functions. Signatures should be updated at regular intervals in order to prevent new attacks and threats.

2.2.3 Intrusion detection system

Intrusion detection is the process of sensing and analyzing system and network events in order to identify possible security breaches in the network and unauthorized access of system resources. A system that performs this process is called an intrusion detection system [190]. Cryptography and firewalls are used to protect the network and system from external attacks, whereas IDS is used to prevent internal attacks (i.e., from an adversary that has already accessed the network or system or that knows the prerequisites). Generally, IDS is comprised of three major modules that are responsible for the entire operations of the system [137]. These are:

(a) Information collection: observe the events, generate and collect information on the observed events

(b) Analysis engine: identify the signs of a breach from the gathered information

(c) Response module: initiate appropriate action against the identified breach

IDS can be grouped into two categories based on the implementation of the employed detection methodology. Host-based IDS (HIDS) and network-based IDS (NIDS) are two kinds of IDSs that are based on implementation. HIDS is implemented on a specific host to observe the events within the specific host, whereas NIDS is implemented on a specific host in order to monitor the network activities that occur where the host resides [138, 190]. IDSs are also categorized into anomaly-based and signature-based detection depending on the detection technique [137, 190]. A signature-based IDS is also known as rule-based or misuse-based. It requires knowledge of acceptable behavior, either in the form of signatures or rules, and classifies an event that did not fulfill the provided behavior as an intrusion. Therefore, the list of acceptable behavior should be updated frequently in order to detect new attacks. Anomaly-based IDSs are used to detect irregularities through observation of events. They are the opposite of signature-based and do not require any prior loading of acceptable behavior.

2.2.4 Access control system

Access control is the process of regulating access requests, either by granting or denying access, to the system resources (e.g., information and services) [190, 191]. A system that performs the access control process is called an access control system (ACS). It is broadly grouped into two classifications:

(a) Discretionary access control (DAC): An ACS that enforces a security policy on the system resources based on the subject identities and allows a subject to change resource attributes or grant privileges to other subjects. The term discretionary refers to the activity of the subject, who have resource access rights, can pass the granted access rights to other subjects by their own volition [190, 191].

(b) Mandatory access control: An ACS that uniformly enforces a security policy on the system resources or objects based on labels that determine sensitivity. “Mandatory” refers to the state that confines the subjects, which have rights, from modifying the security attributes of the resource and from enabling other subjects to access resources [190, 191].

ACSs, particularly DACs, are one of the most commonly used security mechanisms in IT systems. Role-based access control (RBAC) and attribute-based access control (ABAC) are other types of ACSs. In RBAC, security policies are imposed based on the roles rather than on the individual subjects [191]. Security policies are segregated depending on the roles and subjects, and are mapped to respective roles based on their responsibilities. For instance, security policies of the administrator role grant administrative rights to their subjects by allowing them to create, modify and delete objects. Furthermore, it enables administrator users to grant and deny access to objects for users in the other roles. ACSs that mediate access based on the attributes of the subjects and the requested objects are called ABACs. The security policies of ABACs take different attributes (e.g., user, resource, environment, and access rights) to carry out enforcement [191].

2.3 Self-awareness

In psychology, self-awareness can be presented in the concept of “me” and “I”. The former is referred to as objective in nature, while the latter is considered subjective [20, 21, 22]. Objectivity is perceived through the publicly-identifiable features of self-object or self-image and is viewed as collective experiences which are private to oneself and cannot be external (i.e., publicly identifiable). Subjectivity, which concerns self-awareness, will fit perfectly with IT security. As such, system operations are evaluated based on exhibited behaviors that are accomplished through learning experience.

The concept of “self“ was introduced by IBM in 2001 through their manifesto on autonomic computing (AC) [23, 24]. Initially, four concepts of “self” were presented under the umbrella of self-management: self-configuration, self-optimization, self-protection, and self-healing [23]. Later, in 2004, the fifth “self,” self-awareness, was introduced to AC [25].

The self-awareness of a system can be described as its capability to monitor its own state and operational environment to optimize and function accordingly [26]. These systems are called self-aware systems (SAWS). A functional overview of SAWS is given in Figure 2. The crucial aspect of SAWS is their ability to learn and employ the outcome to fulfill the given objectives. Depending on the field of im-

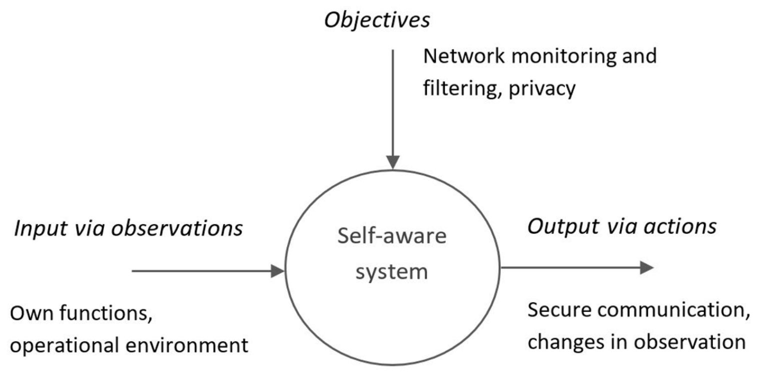

Objectives
Network monitoring and filtering, privacy
Input via observations
Own functions, operational environment
Self-aware system
Output via actions
Secure communication, changes in observation

Figure 2. Functional overview of a self-aware system

plementation, SAWS may be downgraded to function with minimum operations in order to fit with the available requirements, or even upgraded to execution functions, like observing and learning the behaviors of nearby devices.

In a security context, SAWS is used to counter specific threat patterns or groups of attack patterns. SAWS can be employed as agent-based systems and trained to perform certain tasks. Agents will be trained prior to the deployment. It is also possible to use on-field training for agents during execution by providing live system events, but caution must be exercised so that the outcome should exhibit the least number of false positives. The objectives of SAWS will be privacy breach prevention, network restriction, and attack prohibition (e.g., DoS and selective forwarding).

3 IoT and Mobile Devices: Architecture and Basic concepts

Architecture is the very first step that plays a critical role in the design and development of any system. It provides a holistic view of the entire system and helps all involved parties to understand its functions and interdependencies. For instance, the architecture of a home or an apartment defines its physical shape and parts (e.g., walls, floor, pillars, doors, and windows), which are assembled by establishing connections in order to have meaningful and useful space (e.g., living room, kitchen and bedroom). The very same principle is applied for systems as well.

The system architecture is a process that establishes a structural framework that describes its form and identifies the major components and the communication between them [12, 186]. National Institute of Standards and Technology (NIST) refers to architecture as a set of system representations related to physical and logical views that describe elements, interconnections, and relationships at different abstraction levels with different scopes [188]. Institute of Electrical and Electronics Engineers (IEEE) 42010 describes architecture as the "fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution" [187], while Internet Engineering Task Force (IETF) defines architecture as "the structure of system components, their relationships, and the principles and guidelines governing their design and evolution over time" in RFC 4949 [190].

These four explanations agree on the same point, which is, system architecture provides a base for the design and development of a system. The system architecture and its related properties of IoT and mobile platforms, which are necessary for the dissertation, are discussed in a detailed manner in the following subsections.

3.1 Internet of Things

The main purpose of the IoT is to connect every "thing" through an IP-enabled network. The "things" are devices, which are capable of identifying themselves with a unique ID, sense/gather, and exchange of sensed data and control data over the wireless medium. The sensing capability of a device depends on the deployed application. For instance, a heat sensor used to monitor the temperature in a specific area or room. The wireless technology, used to exchange information, also varies, as several technologies are available (e.g., Bluetooth Low Energy (BLE) [74], Zigbee [75], ZWave [76] and LoRa [77]). The IoT devices are comprised of sensors, actuators and interpreters. These devices are limited in computation, storage and energy capabilities. Hence, they are called resource-constrained devices (RCDs) or nodes. The interpreters act as a bridge between the RCDs and external networks. These interpreters are labeled as border routers.

Due to the nature of requirements, IoT operating system (OS) is completely different from desktop and server OS such as OpenBSD [78], openSUSE [79] and Windows [80]. RCDs operate under real-time environments, consume very low power, execute specific tasks, and are expected to produce reliable information. Therefore, the requirements of the IoT OS are reliability, real-time execution, low resource consumption and task specific in nature. There are several OSs specifically available for the IoT applications implementation. Contiki [10, 81], TinyOS [82] and RIOT [83] are among the most popular. Contiki is an open-source, lightweight, event-driven and multitasking operating system developed for low power and resource-constrained embedded systems. It contains several components such as threads, timers, processes, and a radio duty cycle driver. Contiki supports both versions of IP and has a IPv6 over low-power wireless personal area network (6LoWPAN) layer in the protocol stack to support the 6LoWPAN network. It also uses RPL [84] as the routing protocol for 6LoWPAN networks and a Cooja simulator [11] for the simulation purpose. We have used Contiki OS for our research to carry out the experiments using our proposed architecture.

3.1.1 IoT Architecture

Though IoT has vast application areas, it does not have a standard architecture. Several works have proposed layered architectures by addressing the specific application implementations [180, 181, 182, 183, 184, 185, 189]. In [184], the authors proposed a five-layered architecture for IoT and gateway: perception, network access, network transmission, application support, and application presentation. The perception layer will physically connect the IoT devices with the environment and performs information perception and data collection. The network access layer allows sensor devices to form networks (e.g., 6LoWPAN and ZigBee) in order to transmit and exchange their sensed information with each other and with edge routers. The network transmission layer provides connectivity to the external network for sensor networks. The application support layer represents the middleware technology, cloud computation, and database for storage, and performs information processing as per the application requirements. Finally, the application presentation layer provides tools for developing and hosting the applications to deliver accessibility means for the users.

A generic IoT architecture for smart cities is proposed in [180]. Different IoT applications managed through central command that forms the core part of the proposed architecture. This central command is referred to as an integrated information center. In [182], the authors proposed an IoT architecture specifically meant for visually impaired users. It also comprises five layers. The two lower layers comprise the sensor node and gateway, and the two upper layers have background services, storage, and host applications. The middle layer handles the communication between those layers through Wi-Fi and mobile data networks. IEEE has published a standard for an architecture framework for IoT in IEEE 2143-2019 based on the concerns shared across multiple IoT domains (e.g., smart grid, smart building, smart manufacturing, and intelligent transport systems) [189].

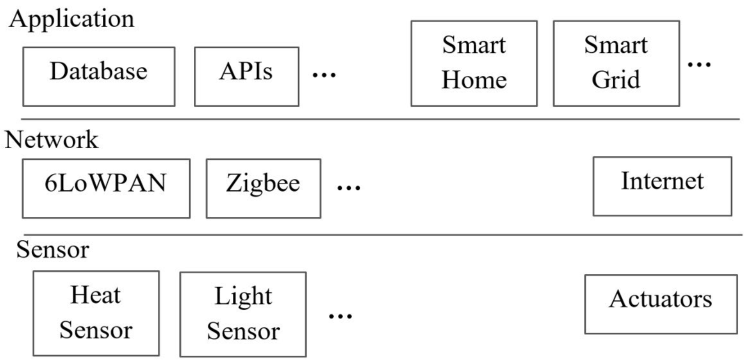


Figure 3. Three Layered IoT architecture

The three common things that exist in the IoT architectures proposed in the literature are sensors, networks, and applications. With this, we can conclude that IoT architecture should at least contain those three capabilities as layers, as is graphically illustrated in Figure 3. The left part of architecture describe the components to build applications and the opposite part show the end result. In the application layer, right part show the IoT applications whereas the database and APIs are the building blocks for application are given in the left part. Similarly for the network layer, networks that enable IoT devices to recognize and establish communication with each other for information exchange are given left part while the network which transfer the RCDs data to server is presented in right part. In the sensor layer, sensors engage in gathering contextual data and execute actions based on the sensed information through actuators. The network layer enables communication between the sensors and other IoT devices (e.g., edge routers) by employing communication technologies. It is responsible for recognizing nodes, formation of networks, processing, transmission, and reception of contextual and control packets between IoT devices and external networks. The application layer contains tools for the implementation and delivery of applications. It also provides means for users to access the services from their devices.

3.1.2 Wireless Technologies, Standards, and Protocols

Since RCDs rely entirely on the wireless medium for information exchange, the employed wireless technology determines interoperability and feasibility in communication among RCDs irrespective of the vendor and hardware. There are several wireless technologies (e.g., Wi-Fi and Bluetooth) available, but most are unsuitable due to their requirements, which cannot be fulfilled by IoT devices. For instance, for surface pressure fluctuation detection, RCDs are expected to operate with battery power for one to three years. In such cases, the use of low power communication and efficient radio transceiver management is necessary. IEEE 802.15.4 [85] and Bluetooth Low Energy (BLE) [74] are targeted for such purposes. BLE specification provides operational guidelines for Bluetooth implementations [86]. IEEE 802.15.4 is a widely accepted wireless technology popularly employed in IoT applications. This dissertation also used IEEE 802.15.4 communication technology in one of the experiments.

3.1.3 IEEE 802.15.4

IEEE 802.15.4 is a communication standard that was defined by IEEE in order to standardize small wireless networks. Small wireless networks that have little or no infrastructure, cover an area of about 10-100 meters, and have a relatively low data rate are called wireless personal area networks (WPANs). The IEEE 802.15.4 standard defines the following:

(a) operational guidelines for L1 (i.e., physical layer (PHY))

(b) operational guidelines for L2, which is the media access control layer (MAC)

(c) low power, low data rate, low complexity, low-cost implementation for WPAN

IEEE has been updating 802.15.4 successively over the past 15 years by addressing new application areas and their functional requirements. For instance, in comparing the 2011 version [87] against the 2020 version [85] of 802.15.4, the latest version addresses the requirements for new application areas, such as rail communication and control (RCC), smart utility network (SUN), and low-energy critical infrastructure monitoring (LECIM).

The main objective of 802.15.4 is to establish a low-rate WPAN (LR-WPAN) with a simple protocol, reliable data transfer, low power consumption, and easy implementation. This standard supports a data rate with a minimum of 20 kbps to a maximum of 250 kbps and can be modified as per the application requirements. Zigbee [75] and WirelessHART are examples of protocol stacks that are defined on the basis of the IEEE 802.15.4 standard [88, 89]. RCDs are categorized into two types under the IEEE 802.15.4 standard:

(a) Full-function devices (FFDs): PAN coordinator or a coordinator in LR-WPAN.

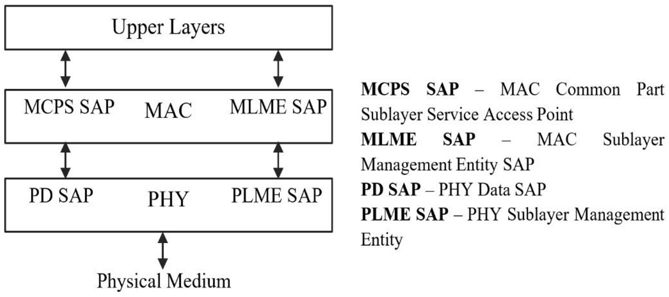

Upper Layers
MCPS SAP
MAC
MLME SAP
PD SAP
PHY
PLME SAP
Physical Medium
MCPS SAP – MAC Common Part Sublayer Service Access Point
MLME SAP – MAC Sublayer Management Entity SAP
PD SAP – PHY Data SAP
PLME SAP – PHY Sublayer Management Entity

Figure 4. IEEE 802.15.4 PHY and MAC layers

(b) Reduced-function devices (RFDs): Act as supporting devices to FFDs and members of the LR-WPAN.

The PAN consists of several FFDs and RFDs. In order to ensure the proper functioning of the LR-WPAN network, it should have at least one FFD as a PAN coordinator. The IEEE 802.15.4 standard defines layered architecture and processing of data packets between those layers. It outlines the procedures for PHY and MAC layers, as illustrated in Figure 4. PHYs are recommended to operate in different frequency bands along with multiple spread spectrums in order to attain different data rates in the same or different frequency bands. Furthermore, different frequency ranges and data rates of IEEE 802.15.4 are given for different regions, as tabled in Table 2.

Table 2. Frequency range, data rate and regions for IEEE 802.15.4

Frequency Range	Data Rate (Kbps)	Region
868 MHz	20	Europe
902-928 MHz	40	US
2400-2483.5 MHz	250	Worldwide

The PHY has a radio transceiver for communication and information exchange with other IEEE 802.15.4 radio networks. It also provides two services: a) *data service*, which handles the transmission of the protocol data unit (PDU) over a radio channel, and b) *management service* that provides services to the MAC layer as commands in the form of constants and attributes to comply with the PHY operations. For example, aTurnaroundTime defines the turnaround time for Rx-to-Tx or vice versa, and aMaxPHYPacketSize specifies the maximum permitted octets for the PHY service data unit (PSDU) [85]. PHY features are energy detection (ED), channel selection, link quality indication (LQI), clear channel assessment (CCA), and activation and deactivation of radio transceivers and packet exchanges [85].

The MAC layer acts as a bridge by providing access between the higher layers and the PHY. The MAC features are beacon management, frame validation, channel access, acknowledged frame delivery, and guaranteed time slot (GTS) management. Additionally, the MAC layer offers provisions for the implementation of the security mechanism. In beacon mode, the MAC uses super frame control and reserves a time slot for transmission. Under beaconless mode, the MAC uses the CSMA channel access without any channel reservation. The MAC layer uses different frames, such as acknowledgment, beacon, data, and command for the execution of its own operations. Similar to the PHY, the MAC layer also provides two services: *data services* and *management services*. The MAC handles the transportation of MAC PDU (MPDU) via the PHY's data service. Management services are used to give management commands to higher layers [85].

3.1.4 6LoWPAN & RPL

LoWPANs are usually powered by IEEE 802.15.4 radio transceivers that were specifically meant for low power and low data rate devices. Hence, routing IPv6 over IEEE 802.15.4 networks possess challenges as both IPv6 [103] and IEEE 802.15.4 have specific operational guidelines that are different in nature. The differences between IPv6 and IEEE 802.15.4 are given in Table 3.

Table 3. IEEE 802.15.4 vs IPv6

Criteria	**IEEE 802.15.4**	**IPv6**
Packet size	127 octets (maximum)	1280 octets (minimum, it can vary based on the applications)
Addressing	Short 16-bit or 64-bit extended MAC	128-bit
Resources	Limited processing power, energy, and memory	No limitation on processing power, energy, and memory
Bandwidth	Maximum 250 Kbps (wireless medium only)	Depending on the medium (E.g., 100Mbps on Ethernet)

The routing of IPv6 packets over low-power wireless personal area networks is called 6LoWPAN, which represents a set of standards defined by the IETF to address and solve the challenges in handling IPv6 packets in LoWPAN. For instance, consider the first case of packet size from the Table 3. In order to route IPv6 packets in 6LoWPAN, several changes such as elimination of common fields, compression and reorganization of protocol stack should be carried out. IETF 6LoWPAN aims to achieve this by introducing RFC 4944 ”Transmission of IPv6 Packets over IEEE 802.15.4 Networks” [90]. RFC 4944 defines the customization of IPv6-related protocols and introduced a new layer in the protocol stack called the LoWPAN layer, which enables RCDs to use IPv6 efficiently and effectively over IEEE 802.15.4 networks. It also allows RCDs to be assigned with unique IP addresses and to communicate with other IP-based networks. Implementation of 6LoWPAN-based applications exists in several fields, for example, home automation [91], industrial automation [92], smart metering, asset monitoring, traffic management, smart office, and personal monitoring [89, 93, 94].

The IETF updated certain sections in the RFC 4944 in order to improve efficiency in handling IPv6 over IEEE 802.15.4. For instance, RFC 6282 [95] proposed to improve the compression format of IPv6 datagrams, and RFC 6775 [96] was introduced to optimize the neighbor discovery in 6LoWPANs. These standards were also updated by RFC 8931 [97] and RFC 9010 [98] respectively.

Protocol Stack

6LoWPAN protocol stack is a modified version of the IPv6 protocol stack and supports only IPv6 packets. It has a new layer, called the adaption layer or LoWPAN layer, to process the IPv6 packets as per IEEE 802.15.4. The comparison of the TCP/IP and 6LowPAN protocol stack is shown in Figure 5.

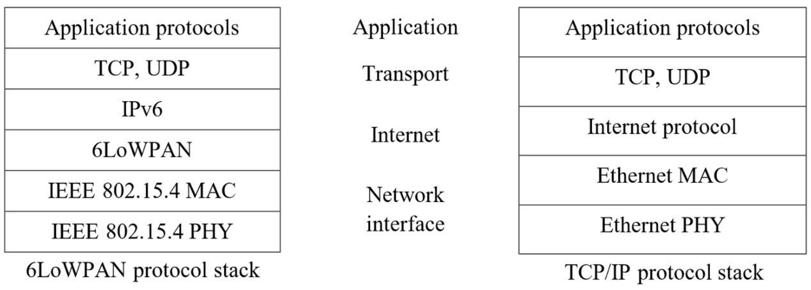


Figure 5. 6LoWPAN and the TCP/IP protocol stack

ICMP protocol is used to exchange control messages for neighbor discovery (ND), route identification, and maintenance. Contextual data is transmitted using either UDP or TCP protocol depending on the requirements and implementation of the application. The LoWPAN layer is used by the router or edge devices, which act as a bridge between external IP networks and the LoWPAN. These router devices are the point of contact for LoWPAN networks, where packets from RCDs are reconstructed into full IPv6 packets to send outside and vice versa. Hence, RCDs within the 6LoWPAN may not be able to use the adaptation layer. The PHY layer handles the information exchange as per the IEEE 802.15.4 guidelines. This includes employing a maximum payload of 127 octets, 250 kbps data rate and an operational frequency of 2400 MHz. The Link layer handles unicast transmission, addressing, and framing. It also handles the exchange of packets to other nodes within its radio coverage. The adaptation layer plays a crucial role in the protocol stack by performing layer two routing, compression, packet fragmentation, and reassembling [88] [99] [100].

Packet Routing

Usually, the packets in LoWPAN networks have to pass through several hops to reach their destination. The process of transferring packets from one device to another device is called forwarding and routing. The routing protocol involves special control messages, which are used to learn about the neighbors and the routes offered by them. This learned information is maintained in a table called the routing table or in a forwarding information base (FIB). The routing table is consulted to determine the appropriate route for the packet and forward it to the next node based on the destination. The processes of routing and forwarding are supported either at layer two (L2) or layer three (L3) due to the placement of the adaptation layer in the protocol stack. The L2 forward is known as "mesh-under" routing whereas L3 forwarding is called "route-over" routing.

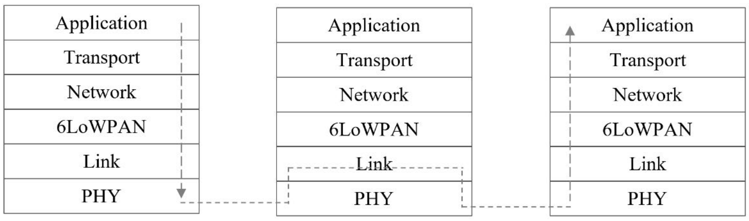


Figure 6. Link layer mesh routing

Mesh-under routing uses a 16-bit short address or 64-bit EUI-64 address to route the packet. The routing process can be executed without or with the involvement of the adaptation layer, as shown in Figure 6 and Figure 7 respectively. The former is referred to as link layer mesh routing and the latter is known as adaptation layer mesh routing. Link layer mesh routing is used in an ISA100 standard [101], in which a protocol responsible for handling the routing operation is provided with an extension to the link layer. In adaptation layer mesh routing, the link layer header contains the source and destination addresses to route the packet towards the destination node's link layer. In this process, the actual destination and source address will be replaced at every hop as the receiving node replaces the source address with its own address and the destination address as the next-hop address. 6LoWPAN introduced a new feature called the mesh header to store the actual source and destination addresses along with the hop count to counter the overwriting of addresses at every hop.

Unlike L2 forwarding, L3 forwarding does not require any additional information, such as a mesh header, for the routing process, and it is a normal routing process. The LoWPAN layer has to perform operations such as decompression, fragmentation

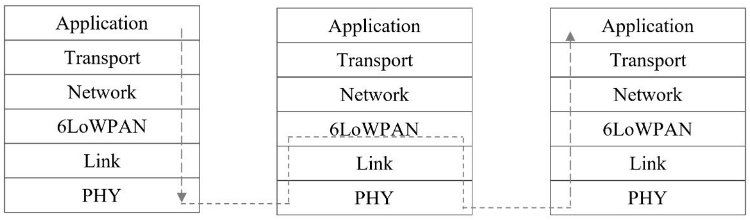


Figure 7. Adaptation layer mesh routing

and reassembly at every hop before sending a packet to the network layer.

Addressing

Two kinds of addressing possibilities exist in 6LoWPAN, these are the MAC address and the network address. The MAC address, also termed the L2 address, is given by the manufacturers as an interface identifiers (IID). The MAC address is created based on the 64-bit IEEE extended unique identifier (EUI). The network address or IPv6 address is a L3 address, which can be assigned or the later part of the 128-bit can be generated from L2 address. In the L3 address, addressing can be carried out based on two scopes (i.e., node visibility): a) *local scope* used to identify nodes locally within the network and b) *global scope* used for the unique identification of nodes in both local and external networks.

Under mesh-under forwarding, a local scope address is sufficient for packet routing within the LoWPAN, but a routable address is mandatory for forwarding packets to external networks. In route-over forwarding, a local scope address is used to route the packets to the RCDs within the radio coverage. A routable address is compulsory for the packets to be routed in both the internal LoWPAN as well as external networks.

Compression

Compression is one of the primary reasons for the introduction of the adaptation layer in the 6LoWPAN protocol stack. Without compression, very few octets will be available for usage of contextual data. The header size of IPv6, UDP and ICMPv6 is 40 octets, eight octets and four octets respectively [102, 103, 104]. The requirement for the TCP header is 20 octets [105]. IEEE 802.15.4 has the maximum permitted size for an outgoing packet at the physical layer at 127 octets. The PHY and L2 headers occupy 25 and 21 octets respectively. Security implementation also consumes additional octets (e.g., the 64-bit AES-CCM implementation consumes 12 octets). This leaves very few free octets available for contextual data transmission. These free octets are sufficient, but they will lead to complexities in transmission and reception, such as fragmentation, defragmentation, and error control, which in turn tax the resources heavily. Hence, the compression of headers is compulsory in order to reduce computations, to enhance performance and improve the operational period.

Stateless and context-based compression are two types of compression that can be performed at the adaptation layer. Context-based compression is a stateful compression, which means that it requires a mutually agreed state or reference between the involved nodes prior to the commencement of compression. Stateless compression is the opposite of context-based compression, as it does not mandate any state or synchronization between RCDs. Two kinds of compression mechanisms exist to reduce the overhead of IPv6 and UDP headers separately.

RPL

The routing over low-power and lossy network (ROLL) workgroup was created by the IETF to evaluate the existing protocols under the low-power and lossy network (LLN) context and to identify requirements for WSN applications [107]. ROLL defined a new protocol to fulfill the identified requirements, such as mobility, scalability, reliability, constraint-based routing, and security [91, 92, 93, 94]. This new protocol is named the IPv6 routing protocol for LLN (RPL) [84].

With RPL, nodes will begin to identify their neighbors and establish routes towards the root, which results in the formation of topology similar to a tree-like structure. The primary feature of RPL is the separation of routing optimization from packet processing and forwarding. A new function called objective functions (OFs) is introduced to handle the routing optimizations. Several metrics and constraints are defined for OFs [108, 109]. This separation allows the applications to define and carry out routing optimizations according to their requirements.

Upon RPL instantiation, nodes will establish a hierarchical natured tree-like topology, which is called a Directed Acyclic Graph (DAG). The DAG is classified into destination oriented DAGs (DODAGs), and each DODAG can act either as an edge router, a convergence point that connects to external networks, or a data collection point. Each node in the RPL network can perform either sensing or routing, or even both of them together [84]. RPL supports three kinds of traffic flow:

(a) **Point-to-point (P2P)**: P2P communication is used when two nodes located far away from each other or within the radio coverage range of each other need to exchange data [84]. Industrial automation [92], building automation [93], and the health IoT are example application scenarios for P2P communication.

(b) **Point-to-multipoint (P2MP)**: The P2MP mode of communication is used when a node is required to collect information from multiple nodes located within the radio coverage. For example, establishing FIB through gathering routing advertisements employs P2MP [84]. It is one of the prime requirements for every IoT application, such as health IoT, smart building, home automation [91], and industrial automation [92].

(c) **Multipoint-to-point (MP2P)**: MP2P is another prime requirement for IoT applications. It is used when nodes, which are located either close or far away from the position of the root, disseminate the observed information towards the root in order to reach servers in the external networks [84].

RPL use control messages for neighbor recognition, route identification, and maintenance. It abides by the rules laid out in the RFC 4443 [104] for the control messages structure. The RPL control message is comprised of a typical ICMPv6 header and a body for both normal and secure messages. The generic packet structure of RPL message is illustrated in Figure 8.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|     Type      |      Code     |           Checksum            |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.                             Base                              .
.                                                               .
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
.                           Option(s)                           .
.                                                               .
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Figure 8. Generic RPL message format

The first 8-bit Type value differentiates the RPL messages from the rest of the ICMPv6 message, and always carries the value 155 [84]. RPL uses four control messages: *Destination Advertisement Object* (DAO), *DAO Acknowledgement* (DAO ACK), *DODAG Information Object* (DIO) and *DODAG Information Solicitation* (DIS). The second 8-bit *code* field is used to identify different RPL messages. The list of code values is given in Table 4. Each RPL message has its payload format and it will be sent in the base field. The *options* field may contain additional information related to the RPL payload. The list of options field values is shown in Table 5.

DODAG Information Solicitation: DIS is a multicast message disseminated to all the nodes within the radio coverage area. It is used to elicit information about the DODAG root from the neighboring nodes. The DIS base object is shown in Figure 9 and may contain *Pad1*, *PadN*, and *Solicited information* in the options field of the DIS base object [84].

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
 - - - - - - - - - - - - - - - - - - - - - - - -
|     Flags     |   Reserved    |   Option(s)...
 - - - - - - - - - - - - - - - - - - - - - - - -
```

Figure 9. RPL DIS payload format

Table 4. RPL messages and corresponding code values

Code	RPL messages
0x00	DIS
0x01	DIO
0x02	DAO
0x03	DAO ACK
0x80	Secure DIS
0x81	Secure DIO
0x82	Secure DAO
0x83	Secure DAO ACK
0x84	Consistency check

Table 5. RPL options field values

options field values	Description
0x00	Pad1
0x01	PadN
0x02	DAG Metric Container
0x03	RIO
0x04	DODAG Configuration
0x05	RPL Target
0x06	Transit Information
0x07	Solicited Information
0x08	PIO
0x09	RPL Target Descriptor

DODAG Information Object: The DIO messages will be disseminated by the DODAG root to the neighboring nodes. The receiving nodes use the instructions provided in the DIO to configure themselves within the DODAG, including parent selection. The DIO payload is given in Figure 10. It may have a *DAG metric container*, *PIO*, *RIO*, *DODAG configuration*, *Pad1* and *PadN* in the options field of the DIO payload. The 128-bit *DODAGID* field contains the IPv6 address of the root node. The *Mode of Operation* (MOP) field determines the role of the node, as either router or leaf, in the RPL network.

Destination Advertisement Object: DAO is a unicast message that is used to propagate destination information that is disseminated by the nodes in an upward direction toward the root. The payload format of DAO is given in Figure 11. The 128-bit *DODAGID* field is an optional part of the payload. DIO's MOP configuration determines the receiver of the DAO. An accepted parent will receive the DAO in storing MOP, whereas, under non-storing MOP, it will be sent to the DODAG root. The

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| RPLInstanceID |Version Number |             Rank              |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|G|0| MOP | Prf |     DTSN      |     Flags     |   Reserved    |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|                                                               |
                            DODAGID
|                                                               |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|   Option(s)...
 - - - - - - - -
```

Figure 10. RPL DIO payload format

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| RPLInstanceID |K|D|   Flags   |   Reserved    | DAOSequence   |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|                                                               |
                            DODAGID*
|                                                               |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|   Option(s)...
 - - - - - - - -
```

Figure 11. RPL DAO payload format

receiving nodes may send acknowledgment if requested or an error is encountered. DAO can use the following options in the options field: *RPL Target*, *RPL Target Description*, *Transit Information*, *Pad1* and *PadN* [84].

DAO Acknowledgement: DAO ACK is also a unicast message, which will be sent by the receiver, either parent or DODAG root, in response to the unicast DAO. The DAO ACK base object is described in Figure 12. Similar to DAO, the ,*DODAGID* field is an optional part of the DAO ACK payload.

RPL messages are disseminated in both directions (i.e., top-down and bottom-up) to learn and establish upward and downward routes. Upward route formation will be initiated by the DODAG root by propagating DIO messages downwards in the RPL network. The receiving node will configure and position itself in the network using the received information, and propagate the same in a downward direction until it reaches the last node. RPL employs a polite gossip policy in the DIO message dis-

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| RPLInstanceID |D|  Reserved   |  DAOSequence  |    Status     |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|                                                               |
                            DODAGID*
|                                                               |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|   Option(s)...
 - - - - - - - -
```

Figure 12. RPL DAO ACK payload format

semination. Under this policy, the frequency and the duration between successive propagation of DIO messages will be reduced as long as the received information and stored information are the same in the receiving node.

In downward route formation, DAO messages will be disseminated by the child nodes in an upward direction toward the accepted parent in the RPL network until it reaches the DODAG root. The requirement of the downward route and the operational mode of RPL depends on the MOP field in the DIO message disseminated by the DODAG root. Under a storing mode scenario, every node (i.e., non-leaf and non-root nodes) should maintain FIB for storing routes learned through DAOs. This creates direct point-to-point communication without any assistance from the root. In non-storing mode, roots store and maintain FIB. A node sends the DAO, and the same information will be propagated by the receiving nodes by appending their destination in the payload. When the DAO finally reaches the root, it will segregate and compute the route from the message. This results in the involvement of the root for routing packets within the RPL, but reduces computation and storage requirement for nodes.

RPL messages are also used to detect and repair the network from inconsistencies (e.g., loop). Local and global repairs are two kinds of repair measures employed by RPL to solve inconsistencies in the formed network. Local repair is a process by which RPL tries to solve the loop by increasing the frequency of control messages within the RPL network. The global repair will be initiated by the DODAG root by changing the version number of the DODAG. This will reset the entire network, and the nodes will acquire different ranks and positions in the resulting network.

3.1.5 Network Security

Security implications are one of the critical hurdles in the implementation of the IoT but security became an after-thought similar to other IT application areas [111, 112, 113]. Since the communication takes place over a wireless medium, it is relatively easy to eavesdrop, capture, and track the conversation between the involved RCDs. With the help of captured messages, disruptions can be introduced into the network, which in turn create a devastating impact on the entire function of the IoT. Depending on the nature of IoT implementation, these disruptions can lead to loss of sensitive data or even loss of life. The implementation of security mechanisms in the IoT is challenging due to limited computational capability, limited storage capacity, and the very low availability of power and data transfer. Additionally, any update on security should be reflected in all RCDs and it is a tedious process to perform manually.

The underlying principle of the IoT network is to forward the received packets until they reach their destination. This very same principle also lays the foundation for most of the attacks, as it is prone to exploitation by attackers. Threats are introduced into the IoT through confiscation and tampering of legitimate nodes or the introduction of a malicious node into the network. Several attacks can be carried out in the IoT, and some that are targeted for the experiment of this dissertation are explained below:

(a) Sinkhole: A malicious node will be introduced into the existing network to attract traffic towards the malicious node through the dissemination of lucrative routing information such as the shortest path [114]. Once the malicious node begins to receive traffic, it will silently discard the packets. It can also engage in other operations such as packet analysis and retrieval of contextual data.

(b) Selective forwarding: A modified version of a sinkhole attack. As the name implies, the introduced malicious RCD will silently drop the packets of specific RCD rather than entire packets originated from all RCDs [114]. When compared to sinkhole attacks, selective forwarding is hard to detect, especially in mobility.

(c) Sybil: A malicious RCD will present itself as a legitimate node through impersonation [114]. In a Sybil attack, this impersonator will be placed in different regions within the network, away from the position of the original node.

(d) Packet flooding: It is a form of denial of service (DoS) attack, which aims to keep the RCDs occupied with unnecessary activities. This ultimately results in a draining of resources and prevention from the execution of legitimate functions. For instance, the HELLO flood attack [114, 115], in which malicious nodes flood the network with HELLO request messages to force the legitimate nodes to select the malicious one as a parent or to break security.

Attacks are broadly classified under two categories: external attacks and internal attacks. External attacks are attacks that originated from the outside to enter the network without having any prior knowledge of it (e.g., employed security mechanism and frequency of message exchanges). Attacks, such as packet flooding fall under this category. Internal attacks are carried out with proper knowledge of the network (e.g., keys used in encryption). A legitimate node will be confiscated, reprogrammed, and reintroduced into the network to disrupt operations. Sometimes, people involved in the implementation can also launch internal attacks due to the conflict of interests. Selective forwarding and Sybil attacks are examples of internal attacks.

The method of attack implementation determines the severity of the attack, and the complexity of its detection and mitigation. A single attack against the IoT network may be detected easily through network observation, but when two or more attacks are mounted simultaneously, detection becomes harder [114]. In addition, detection and removal of a single attack will also overcome other attacks in multiple attack scenarios [116]. The IEEE 802.15.4 standard did not specify any security mechanism for the protection of LoWPAN, but it recommends the usage of encryption at the L2 level and also provides dedicated message formats for handling messages securely. Several implementations of encryption standards, like advanced encryption standard (AES) [117], elliptical curve cryptography (ECC) [118], and RC5 [119], are publicly available as libraries, such as Crypto++ [120], java cryptography architecture (JCA) [121], and Pycrypto [122]. Similarly, numerous intrusion detection systems (IDS) have been implemented for IP networks [123, 124, 125, 126]. Unfortunately, they cannot be used directly due to the computation and memory requirements. Hence, they should be optimized for IoT implementation.

Contiki OS offers security at the L2 and application layer through the implementation of IEEE 802.15.4, TSCH, and DTLS, respectively [127]. DTLS is implemented through the customization of TinyDTLS [128] specifically for Contiki OS. A network security mechanism was proposed for Contiki in [129] by offering three modes of security: ContikiSec-AE (authentication with encryption), ContikiSec-Enc (confidential), and ContikiSec (authentication). It requires a 128-bit network-wide master key, which any compromised node can reveal. This in turn exposes the entire network. In [130], 6LoWPAN security was proposed for Contiki with two modes of operation: adaptable pairwise key establishment scheme (APKES) and easy broadcast encryption and authentication protocol (EBEAP). AKPES allows plugging of a cryptographic mechanism into the implementation and supports three different key establishment schemes: Blom's scheme [131], pairwise keys [132, 133] and LEAP+ [134]. The effectiveness of this implementation is determined by the cryptographic mechanism used in APKES. SPINS [135], a set of security protocols for sensor networks, was proposed and implemented in TinyOS to provide security through encryption. It has two building blocks: secure network encryption protocol (SNEP) and uTESLA. The latter guarantees the authenticated broadcast, and the former ensures authentication, confidentiality, and data freshness. It guarantees the preservation of keys in compromised nodes but does not handle attacks such as DoS. The employment of encryption alone may not detect security breaches as long as exchanged packets abide its procedures. Furthermore, software implementation of the encryption mechanism affects the throughput of the contextual and encryption data [136].

Cryptographic mechanisms prevent external attacks, but they may be completely ineffective against internal attacks due to the adversary's knowledge of the prerequisites. In such scenarios, an intrusion detection system (IDS) plays a crucial role. IDSs are classified based on their implementation and detection techniques. HIDSs and NIDSs are two kinds of IDSs based on implementation. Misuse-based and anomaly-based are two categories of IDSs that depend on the employment of detection techniques. IDSs are generally designed for normal IP networks and IT devices and cannot be implemented in the IoT due to limited resource availability. For instance, if NIDS were implemented on a dedicated RCD to observe specific clusters in the IoT network, this would result in the rapid draining of resources (e.g., battery) due to the continuous listening, capturing, and processing of network activities. Therefore, they should be optimized according to the requirements of the IoT.

Trust-based routing or trust-aware routing is other mechanism proposed to detect malicious-natured nodes in the WSN. In trust-aware routing, RCDs engage in quantifying the trust factor of their neighbors based on the exhibited behaviors and perform voting to assess the trustworthiness of an RCD. Any node that received a low rating will be avoided for routing packets. This approach uses a broadcast type of communication for voting and requesting neighbors' information, and covers a minimum of 2-hop neighbors.

A detection model to prevent insider attacks is presented in [139]. The model consists of four phases: collecting information, filtering collected information, deducing outliers, and voting. In the first phase, every node should monitor its neighbor node and gather data. The collected data will be filtered to remove false information if introduced by the adversary. Mahalanobis distance is used to detect the existence of the outlier, and finally, the list of outliers will be confirmed through a majority vote by neighbor nodes. The detection model is evaluated statistically without any proper network emulation. There is no information related to the resource requirements, though it relies heavily on computation and broadcasting messages. The author claims that the model yielded very low false positives without any discussion on gateway involvement and implementation feasibility.

A hybrid IDS called SVELTE is proposed in [140]. In SVELTE, tasks are classified based on the resource requirements. The least resource-intensive tasks are implemented in the nodes, whereas computation and power-intensive tasks are implemented in border-router. Nodes will perform monitoring and send network-related information to the router. The border router will direct the activities of the network based on the received information. In [141], a generic algorithm-based IDS is presented. In the IDS presented, cooperation among the nodes is essential for the proper functioning of the IDS. Every node is loaded with an alert module that is used to raise an alert whenever an unintended activity is noticed. Furthermore, nodes also contain a set of one-way hash keys from the preassigned key that is loaded in external storage. Nodes engage in voting by publishing their key to determine the trust factor of their neighbor nodes. This process involves broadcasting messages up to two hops.

A real-time intrusion system is implemented in [142]. In this IDS, nodes will monitor and gather the activities of their neighbors, and send them to the border router. The border router will analyze the received information to determine the threat's existence. A similar system that uses border-router to detect attacks is given in [116]. A new type of ICMPv6 message called the heartbeat message is introduced in this work. The heartbeat message will be disseminated periodically from routers to nodes to verify the node's existence and to thwart attacks like selective forwarding. In addition, a provision to isolate malicious nodes is provided in the form of blacklist or whitelist content. A network-based IDS is presented in [144]. Selected nodes will be programmed with NIDS, and positioned within the network to cover the entire network. These selected nodes will function as watchdogs to observe the conversation among the nodes. They are also loaded with rules that help to detect unintended activities. Different rule sets can be loaded in different watchdog nodes, and these rules can be updated over the air. Another NIDS-based approach is proposed in [143]. In the proposed approach, the network is divided into clusters, and each cluster is assigned with a cluster head. Cluster members communicate with their respective heads and exchange information about their neighbors. The cluster head, which contains the implementation of NIDS, is responsible for the detection of malicious activities from the received information.

3.2 Android Mobile Platform

The embedded OS targets specific tasks such as making and receiving calls, exchanging messages, and powering mobile devices. Over the past decade and a half, it evolved greatly in terms of computational performance, functionality, user experience, storage, and accessibility, transforming into smartphones powered by multitasking OSs. Smartphones have become widely popular, and are the most preferred device for connecting people across boundaries due to their usability and comfortability factors. These mobile devices encourage users to use several services in the form of applications, to generate, store, and exchange information, including sensitive details such as financial transactions, personal information, and private events, over different network technologies. There are several OSs that are specifically engineered for smartphones. Android [6], iOS [145], Sailfish [146], KaiOS [147], Fire [148], and LineageOS [149] are among them. The most popular mobile OSs are Android and iOS. The former has been developed and maintained by Google and the Open Handset Alliance, while Apple holds the proprietary rights for iOS.

The OS empowering mobile devices have similar computational and operational capabilities to the desktops and laptops OS, as they were derived from the same base. Thus, they are vulnerable to similar exploitation. Furthermore, smartphone operations are based on subscriptions (i.e., payment of a certain fee to access services such as internet, calls, and messages). Hence, an OS attack can also lead to monetary loss and privacy violations. Due to the popularity and the availability of source code for customization, Android is selected for the dissertation work and is discussed in the following sections.

Android was founded in 2003 and was acquired by Google in 2005. The source code of Android OS is freely available under the name Android Open Source Project (AOSP) [150], but products such as Google Play [151] and Google Mobile Services [152] have been proprietary to Google.

3.2.1 Architecture

The Android platform is entirely based on the Linux kernel [153]. It is customized to offer features like power management, binder, process memory allocator, and asynchronous shared memory. An overview of the Android architecture is illustrated in Figure 13.

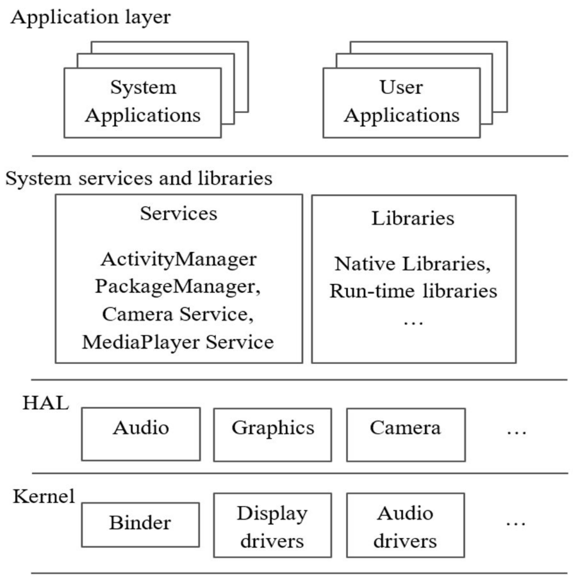


Figure 13. Android architecture overview

The lowest layer in the architecture is the kernel layer. It is comprised of essentials (e.g., device drivers for camera, audio, and radio transceivers). The hardware abstraction layer (HAL) provides interfaces for the hardware vendors. Using HAL interfaces, hardware vendors write their implementations for the integration of their hardware products with Android without affecting the entire platform [154]. The application layer is the place where the device users will install and use the applications of their choice. This layer also host the system applications that comes as a part of the Android. The system services and libraries layer consists of services and libraries that are essential for the applications to execute their intended functionalities. It contains native and runtime libraries (e.g., Android Runtime (ART)), and assists applications through the application programming interfaces (API) to communicate with hardware and other system resources. The responsibility of the system services and libraries layer also includes handling the life cycle management of applications and other services. In addition, it also manage the restriction enforcement on the resource accessibility of applications.

An integrated development environment (IDE) is available for Android application development [155]. Android provides a software development kit (SDK) for the development of applications. The SDK offers libraries and exposes APIs so that application developers can access the system resources and perform operations using them. Android applications are developed using Java and Kotlin languages. C/C++ is also employed for application development through the usage of Java Native Interface (JNI). Google Play [151] has been the official repository of Android applications, where users can download and install applications of their choice. Android charges $25 as a one-time payment during developer registration, which allows developers to publish the applications. Android categorizes applications under three clusters in terms of revenue generation: free, paid and in-app purchase. As the name implies, free applications are free and do not require any payment to download and use the application. Users purchase applications clustered under the paid category before the installation and usage. In-app purchase is a combination of both free and paid applications. In-app purchase applications do not require money for download and installation but do not offer full application features for users. To unlock the features, a user must pay for the services and the payment terms (e.g., a one-time permanent or monthly subscription), depending on the application. Google Play is estimated to have more than 2.5 million applications [156, 157].

Inter-process Communication (IPC) is used to handle the communication between applications, application components, and processes. Android uses a binder for IPC purposes that is derived from OpenBinder [158]. Binder operates as a client-server model under the principle of thread migration. Kernel driver creates and exposes the binder and will identify the process that initiates the IPC call through the process identifier (PID) and user identifier (UID). All the migration tasks (e.g., tracking of object reference and parameter marshaling between processes) are handled by the binder kernel.

3.2.2 Security implementations in Android

Android inherits its user resource isolation and process isolation features from Linux, as it is a Linux-based OS. User resource isolation is modified and extended to the application level. Each application will receive a UID after installation through this modification. The application also executes under the very same UID that was assigned after installation. Apart from the UID, the application will also receive a dedicated directory for the storage and management of its resources. Access to the other files or resources is proscribed unless explicitly permitted. System applications and daemons are owned by the system or root, and they are executed under the assigned pre-defined UIDs. As a result, Android achieves application sandboxing (i.e., isolation at the process level and file level). The isolation achieved through restricting the application activities through the prevention of access to any resources other than its own defeats the purposes of the OS itself. Hence, Android introduced access rights in order for the applications to access resources (e.g., data and hardware features) to overcome the isolation barrier. This is called a permission model.

The permission model allows applications to use resources through permissions. Applications required to access resources should acquire proper permission before the usage of the resource. All the permission requests should be stated explicitly in the AndroidManifest file. Under the permission model, Android offers over 150 permissions to regulate resource access [159]. It also enables developers to define their permissions to enforce access restrictions on the critical resources of their applications [160]. The Android framework handles the enforcement of access verification to guarantee that the applications do not access non-permitted resources during the runtime. Few permissions are mapped directly to low-level OS control that are not monitored by the Android framework. Permissions are classified under three categories [160]:

(a) *Normal* - Permissions allow access to data and the performing of actions, which are least considered or not considered as a risk towards the privacy of users and other applications.

(b) *Dangerous* - These permissions allow applications to access restricted data (e.g., location and personal information) and to perform restricted actions, which possess a high risk to the privacy of the users and other applications.

(c) *Special* - This is defined by the platform and original equipment manufacturer (OEM) corresponding to the particular operations of the application to protect access to critical actions and resources.

Another group of permissions called *signature* permissions also exists in Android. Permissions required by an application are granted during installation and cannot be revoked in the earlier versions of Android. The granting of all permissions during installation was changed from version 6 by introducing a feature for granting and revoking permission at run-time. This feature is named runtime permission. It enable users to grant or revoke permissions to any application before the execution or through settings.

Permissions are divided into two groups: *install-time* and *run-time*, for this implementation. *Install-time* permissions are granted during installation and cannot be revoked by the user. Permissions under *normal* and *signature* classifications fall under this group. *Run-time* permissions are comprised of *dangerous* permissions that the user can revoke or grant at any time.

SELinux was introduced to Android in version 4.3 as SEAndroid, to enhance security by enforcing mandatory access control at the kernel layer. SELinux implementation mode was set to enforcement from permissive mode in version 5. In version 8, SELinux policies were split into two groups: *platform* and *non-platform*, to perform policy updates and maintain compatibility. The platform group was further separated into two subgroups: *platform private* and *platform public*, to export certain attributes and types to the non-platform policy. The list of security-related enhancements that were introduced into Android from versions 4.3 to 13 is tabled in Table 6 [161].

Table 6. Security-related enhancements for Android version 4.3 through 13

Version	Enhancements
4.3	Introduction of SELinux and permissive mode of operation Authentication of Android debug bridge (ADB) with RSA key pair since 4.2.2
4.4	SELinux set to enforcing mode
5.0	SELinux enforcing mode to all domains Guest and user restricted profiles Full disk encryption by default and improvement through scrypt Updated cryptography for HTTPS and TLS/SSL
6.0	Runtime permission Verified boot Clear text traffic restriction USB access control
7.0	Strict enforcement of verified boot File-based encryption (FBE) APK signature scheme version 2
8.0	Verified boot through Android verified boot (AVB) Install unknown apps restriction
9.0	Biometric support for authentication FBE for adoptable storage Per-app SELinux sandbox
10.0	Face authentication Bounds sanitizer in Bluetooth and codecs OEMCrypto API version 3
11.0	Generic Kernel Image (GKI) architecture API quotas OEMCrypto API version 16
12.0	MAC randomization Approximate location accuracy WPA3 Hash-to-Element (H2E)
13.0	APK Signature Scheme version 3.1 Shared UIDs deprecation Android virtualization framework

The incidence of attacks and security breaches in mobile devices dates back to a virus found in Symbian OS in 2004 [162]. The malware used multimedia messaging services (MMSs), Bluetooth, and internet connections to infect the devices and spread the virus through messages. Some viruses were designed to target desktops through USB connections. Adversaries began to target Android, as it was the dominating platform of the time, occupying the largest share of the mobile market in 2012. The fundamental nature of mobile device operations (i.e., their subscription and payment model), became a vulnerability exploited by adversaries. Under the guise of normal applications like media players and games, applications with malicious intention targeted the fundamental nature of mobile device operations. These applications sent messages to premium numbers that resulted in monetary loss for the users. The sophistication of the malicious programs begins to increase when Android raised the security of the platform.

3.2.3 Proposed security approaches from the research community

Android also attracted the interest of the research community, which investigated the possibilities of exploitation and proposed solutions accordingly [163-179]. CRePE, an access control system with policy enforcement for Android is given in [164]. The enforcement of a policy depends on the environmental contextual information: time, GPS, and Wi-Fi. Since its context is based on time and location, CRePE can be used to restrict or allow actions based on the location or nature of the operation. XML language is used to write policies, and separate commands and keywords are specifically defined for this purpose. Policies can be administered locally through the local administrator component, and they can be administered remotely via Bluetooth, QR codes and SMS. CRePE uses Android permissions to enforce policies and follows static policy enforcement procedures. The policy is required to be written manually in XML, but the status modification of policies can be performed at runtime using given interfaces. The degree of device control activities is unclear regarding several activities like application installation, network association, and the limited action in policy refinement. Furthermore, information regarding the local administrator component, such as usability and availability, is unclear.

A security framework called TrustDroid is used for domain isolation to prevent the unauthorized communication and data accessibility of applications based on different contexts [165]. It restricts inter-component communication (ICC) at the middleware layer by employing mandatory access control at the same layer. In addition, TrustDroid uses TOMOYO Linux as a kernel layer mandatory access control to restrict the IPC, file system, and network. IT uses policies combined with location information from GPS and Wi-Fi to perform isolation. A policy-based framework called MOSES is presented in [166, 167]. Policy enforcement for the software isolation of data and applications is achieved through virtualization. MOSES stands for mode-of-uses security separation. It combines TaintDroid [168] architecture with virtualization at the Android middleware layer for isolation and supports the creation of several security profiles (e.g., GPS and Wi-Fi) based on contextual sensor information to enforce different kinds of restrictions and switch between them dynamically.

In [163], the authors proposed a security framework, named XManDroid, which extends the monitoring mechanism of Android to detect and prevent application-level privilege escalation attacks at runtime based on a system-centric system policy. XManDroid monitors all communication links between apps and verifies them against a set of policy rules. It also tracks entries written on shared channels and filters access to these channels. According to the results observed from experiments, XManDroid requires precise policy engineering to minimize false positives. In [169], the authors improved XManDroid by additionally validating ICC using an intent-tagging-based call chain technique and by enforcing kernel-level mandatory access control. These improvements use a high-level policy language at the kernel level and adopt TOMOYO Linux for access control enforcement at the kernel level. In addition, a dynamic policy mapping is developed to bridge the semantic gap between policies at the two levels. In both XManDroid and its extension, it is difficult to outline precise security rules for regulating all possible interactions among installed apps on a device. In addition, some desired inter-application communication and ICC may be rejected due to the proposed strict policy enforcement.

FlaskDroid is presented in [170] as a mandatory access control for Android. FlaskDroid enforces restrictions in both the middleware and kernel layer. FlaskDroid policy language is inspired by SELinux and is specifically designed to include the middleware layer of the Android architecture. The implementation of FlaskDroid is based on SEAndroid. ConXsense, a framework for context-aware access control based on automated context classification is proposed in [171]. ConXsense uses a probabilistic approach for sensing different contexts and machine learning for the automatic classification of context. ConXsense is implemented on top of flaskdroid [170] architecture as it provides an access control system based on SEAndroid. Two uses cases, a) device misuse protection with dynamic lock and b) sensory malware protection, were analyzed using the framework.

In [172], a framework for protecting data through data classification and policy enforcement in mobile devices has been proposed for enterprise-level security. The framework has four major components: sensitive data isolation, policy formulation, policy testing, and policy execution. It concentrates on securing data by creating separate blocks but it does not reveal any information on threat realization. Several components are specified to write the policy, including the language. The ample amount of policy-defining languages causes a considerable burden on the system administrator. The allocation of individual components to deal with different sets of resources can make the system more complex and might require additional skills to understand and maintain the policies. In addition, with the exception of the data isolation part, the rest of the part can be related to the software development life-cycle model. Since there is no information about implementation details, it is not clear whether the framework is implemented as dynamic memory hooking or OS customization.

In [173], a security framework named Android Security Module that exposes a programmable interface for defining new reference monitors for Android is presented. The security features provide APIs to implement either resource accessibility control or data mocking. Using APIs, one can restrict accessibility by writing apps that can act as reference monitors. Android Security Module does not provide any access control features by default, though the framework hooks almost all system services and components. Another framework called Android Security Framework is presented in [174]. Android Security Framework performs the same operations as [173] but also assures existing Android security provisions and an inline reference monitor. These two frameworks do not implement any control by default, even if it is available within the OS. Applications must be developed and installed in mobile devices to realize the resource restrictions. The enterprises have to model the security system architecture and develop apps accordingly due to the unavailability of provision to control the server when needed. Individual users who do not have programming knowledge must rely on the applications developed by third-party sources to fulfill their objectives, and this may expose devices to privacy and security threats. Finally, transparency is not guaranteed by default in the system. This makes individual users completely incapable of understanding the activities of the security applications.

All the above approaches require modification of the Android platform for implementation. The authors in [175, 176, and 177] proposed a different approach that does not need platform modification. DeepDroid, a dynamic security policy enforcement mechanism targeted for enterprise usage, is presented in [175]. The enforcement of policies based on location information in DeepDroid also requires root privilege for its operations. DeepDroid is implemented through the dynamic memory instrumentation of certain critical processes and enforces policies through it. This results in the implementation of enforcement without platform modification. DeepDroid lacks granularity in context as it uses only location information and it lacks granularity in policies. In [176, 177], the authors proposed the approach of opening and repacking APK files after the introduction of reference monitors. These reference monitors are used to observe the application behavior and restrict them from engaging in malicious activities such as sending premium SMSs or accessing malicious IP addresses. There are several drawbacks to this approach. Application original signature will be broken during the repacking process and the management of signatures for each application to retain the original signature will incur additional operational overhead. Users may possibly install an application from other sources that offer applications without reference monitors, which results in abnormal application execution. Additionally, if the placement of reference monitors does not cover all APIs, the occurrence of security breaches is highly likely.

3.3 Problems in current approaches

The primary issue with the current approaches is security, which has never been considered as part of any system from the design stage. During or after system implementation, security mechanisms will be employed as tools to protect the system functions. They must update some part of their own information base or else may face additional vulnerabilities apart from the vulnerabilities of the implemented system. It is understandable that already-implemented and running systems cannot be changed or redeveloped due to factors like cost, time, human labor, and skills, but the systems that will be developed or undergo an upgrade in the near future should be changed. The problems that should be addressed during the change are as follows:

Architecture: In the design stage of any IT system, any one of the two architectures (i.e., centralized or distributed) will be employed. The former is the oldest system architecture used in the design process, especially for decision-making and communication purposes. Client-server is a fine example of centralized architecture. In a client-server model of operations, a single server will configure and maintain multiple clients simultaneously. Decisions are established on the server and then disseminated to the clients. In a centralized architecture, manageability of clients becomes tedious when they are in huge numbers. In addition, any threats or attacks on the server will bring down or cause a severe impact on the entire operations and may be transferable to the connected clients as well.

To counter this issue, centralized authority is dispersed, resulting in the formation of distributed architecture. For example, in network security, the distributed architecture enables the creation of clusters (i.e., multiple segments) from huge networks, and each of them is assigned with cluster heads that act as watchdogs and regulate the network activities. In some advanced networks, the cluster head role is transferable and assigned to different entities in the same network after predefined or agreed upon intervals to counter the attacks targeting cluster heads. IT has evolved into the IoT, which represents the interconnection of every device in cyberspace including IT systems, even home appliances like washing machines and refrigerators. In this scenario, connected devices should react to changes themselves, particularly against threats, in order to protect or reduce the damage, at the least temporarily, without waiting for the instructions, and the architecture should be changed in order to accommodate it.

Cooperation among security mechanisms: Security mechanisms have been used as tools to establish layers in order to secure system functions and resources. The different security tools are discussed briefly in section 2.1. Hardly any form of coordination or communication exists among them during execution. For instance, cryptography provides a secure communication channel for information exchange and protection is guaranteed as long as employed keys are computationally safe. Firewalls with an IDS/IPS protect the network by preventing unauthorized access and abuses from unknown sources and inferring suspicious behaviors from the network traffic. Even though the firewall becomes aware of the suspicious behaviors and employed countermeasures within its capability, it cannot coordinate with cryptography to forego existing keys and reestablish new keys in order to strengthen the communication channel. This should be performed manually or using software tools. In these cases, the likelihood of vulnerabilities introduced by the software tools or manual error cannot be ruled out. Hence, the lack of cooperation among the security mechanisms should be addressed.

Initiator and responder activity: A widely used activity in IT systems, particularly in communication and execution. For example, under the IoT network communication, routers, cluster heads, and roots use a polling mechanism that sends a message towards the connected devices within the network or cluster to verify the status (e.g., available power and surrounding conditions). The received devices will respond to the initiator with the requested status. In the client-server mode of operation, the clients execute certain functions as a response to the request from the server. This may cause serious issues when combined with the lack of cooperation among security mechanisms concerning the IT infrastructure.

Network awareness: Since every device is connected and has some central figure in the form of any kind of IT system, the central figure's requirement for network awareness becomes an essential trait which is missing in the current approaches. This requirement is also related to architecture. It will enable the central figure to understand the dynamics of the network and the activities of the connected devices without any assistance from outside of the network or tools. In the fully automated scenario, a central figure can create clusters, provide limited autonomy within the clusters, and establish communication with cluster heads directly rather than every cluster member. This will enhance the competencies of network management and help to achieve self-management. Network awareness helps central figures to approximate the location of a particular connected device through their neighbors without the requirement of having location-based hardware. This will become opportune in the deduction of locating the threat origin or the affected devices in the network.

Regulation of activities: This is an important and crucial requirement in the connected cyberspace environment. It is closely related to the architecture and cooperation among security mechanisms. As stated in the previous case, a central figure will exist in the IT network. Regulation of activities occurs as an automatic reaction against some kind of unusual behavior by the connected devices without waiting for the central figure or cluster head commands. This reactive nature requires the armament of low-level devices in the network with some kind of security mechanism, as per the available resources, which in turn empowers them to make and enforce decisions that will reduce the impact of unusual behavior from causing problems in the network. Of course, the countermeasures may be trigged due to false positives during the determination of abnormal incidents, but false positives can be reduced through various means (e.g., a collective approach and profile).

Apart from the above-listed problems, other issues, such as operational environment awareness, observation links, and learning also require attention. Operation environment awareness assists greatly during the regulation of activities, as it enables the connected devices to understand the patterns of activities that have been going around and within the devices. It is also related to learning to a certain extent, as understanding requires some degree of learning. Observation links, in a sense, are points that allow devices to monitor activities and be able to deduce the required information, which is useful for learning or enforcing regulation.

4 Natural and Biological Inspiration in Science and Technology

Nature refers to the entire planet along with life forms that exist on the planet. Nature experiences numerous problems and develops appropriate solutions. For instance, trillions of insects that cause damage to vegetation and ecosystems have been kept in check by billions of birds and other species. The solutions presented by nature provide ideas that stimulate humans to solve problems or execute tasks efficiently. In several instances, humans have derived solutions inspired by nature; birds' collisions with windows are one such instance. The modern concept of bringing natural light into buildings and homes through the placement of very large-sized transparent glass windows and walls has resulted in the death of birds at a large scale. Some estimates state that hundreds of thousands to millions of birds have been dying every year due to colliding with glass windows [27, 28, 29, 30, 32]. Attraction towards light and reflection, and the inability to notice the glass during flight are the primary causes for collision. This resulted in the design of glass inspired by spider webs [33, 34]. Spiders have the ability to decorate the webs with ultraviolet (UV) reflections in order to attract prey [31, 35, 36] and these UV orbs are visible to birds [31, 33].

4.1 Swarm Intelligence

In the information technology (IT) field, nature-inspired mechanisms like swarm intelligence have helped scientists solve problems (e.g., shortest path). Beni et al. introduced swarm intelligence in the robotic system [37]. Swarm intelligence takes inspiration from the activities of life forms, especially insects and animals. For these life forms, intelligence and problem-solving abilities are limited to the individual level, but as a group, even complex problems that are beyond individual capacity can be solved easily. The swarm intelligence of ants and honeybees has been researched extensively over the decades. For instance, honeybee-inspired optimization and ant colony optimization are derived from the swarm intelligence of the respective insects. Ants have the ability to determine the shortest path to reach their destination without any visual assistance or the presence of any centralized authority [38, 39]. This behavior has been studied and used to solve different problems, like those of traveling salesman, job-shop scheduling, and resource-constrained project scheduling [40, 41, 42, 43].

4.2 The human immune system

Biologically inspired mechanisms like the human immune system, have also been researched for use in IT. The human immune system (HIS) is a complex network comprised of numerous organs and cells [44]. The primary responsibility of the HIS is to defend the human body against any invading harmful parasites and microbes, called pathogens. Lymphocytes are special cells produced by lymphoid organs that are positioned in different regions within the body. They are responsible for the activities of the HIS through B cells and T cells [44]. These cells derive their name from their place of origin. T cells are developed in the thymus, and B cells originate in the bone marrow. B cells secrete antibodies that work against pathogens, while T cells kill the cells' body infected by the pathogen or produce a cell surface that helps other infected cells to kill the pathogen. The defensive activity of the HIS inspired the same in IT security [45, 46, 47].

4.3 The human nervous system

The human nervous system (HNS) is another biological system that forms the core of the dissertation work. The operations and the organs involved in the HNS are discussed briefly in this section.

The human nervous system stretches across the human body through nerve cells called neurons. Through neurons, the HNS allows the human body to interact with external and internal environments. Hence, it is referred to as a communication system. It also regulates and controls the activities of the body (e.g., walking, breathing, speaking, and learning) both actively and passively. Therefore, it is also referred to as a regulatory system or control system [48, 49, 50]. The HNS is divided into two subdivisions, the *central nervous system* (CNS) and the *peripheral nervous system* (PNS), based on their placement in the human body. It consists of organs like the brain, spinal cord, ganglia, and nerves. The HNS has several billion neurons, and each of them has a cell body and extensions. A neuron is graphically presented in Figure 14.

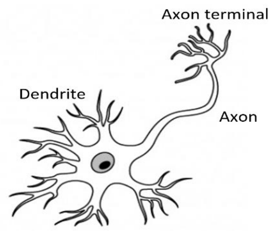


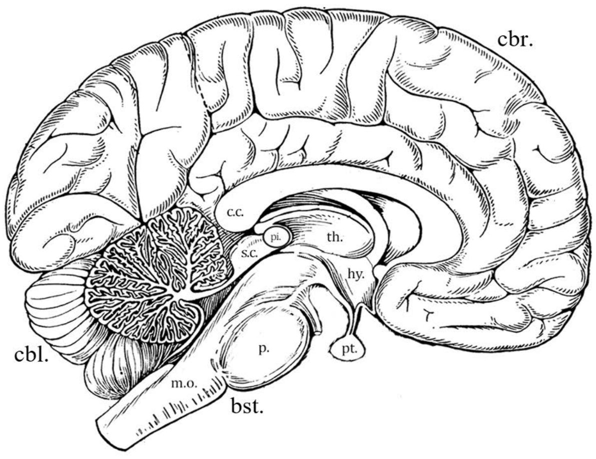
Figure 14. Structure of a neuron

Each neuron can have multiple dendrites extension, usually short in length, acting as an antenna to capture the signal from nearby neurons and an axon to carry the signal away from the cell body [48, 51]. Neurons are classified as afferent, efferent, and interneuron, depending on the direction of the signal movement. Afferents, also known as sensory neurons, transfer signals from PNS receptors to the CNS, whereas efferent neurons transfer impulses from the CNS to effector organs, such as glands and muscles. Efferent neurons are also referred to as motor neurons. Interneurons exist only in the CNS and connects afferent and efferent neurons links. The various activities of HNS can be broadly classified under three general functions:

(a) *Sensory function*: The sensory receptors are responsible for the detection of changes in the external and internal environment. It will monitor and capture the changes, which will be listed as stimuli. For instance, the receptors can sense changes in sound, light and temperature from the external environment. In the internal environment, they can sense the changes in blood pressure, the concentration of gases, and water in the bladder. These detected changes are converted into electrical or chemical signals and directed towards the CNS.

(b) *Integration function*: The received signals are combined, assessed, and stored or discarded from the memory. Finally, a decision will be made against every received signal. The spinal cord and grey matter in the brain perform the integration function.

(c) *Motor function*: The execution of the decision made by the nervous system against the received signal. The decisions are sent to the motor nerves in the form of signals to carry out activities such as the expansion of muscles, secretion of hormones, and movement of the body parts or whole.



cbr.	Cerebrum
cbl.	Cerebellum
bst.	Brain stem
m.o	Medulla oblongata
p	Pons
pt.	Pituitary gland
c.c.	Corpus callosum
pi.	Pineal gland
s.c.	Superior colliculus
th.	Thalamus
hy.	Hypothalamus

Figure 15. The human brain

4.3.1 The central nervous system

The central nervous system is comprised of the brain, spinal cord and the nerves in those organs. The organs of the CNS are critical for the HNS, and thus, they are secured safely inside the skull and the vertebral canal of the spine, respectively.

The brain is the most complex and important part of the human body, where almost every activity has been initiated and regulated. A diagram of the brain is shown in Figure 15. The brain can be described in three subsectors: the *forebrain*, the *midbrain* and the *hindbrain* [50, 52, 53, 54]. The forebrain consists of the cerebrum and the diencephalon. The cerebrum is the topmost part of the brain, divided into two hemispheres. It occupies around 85% of the brain and is responsible for intellectual activities like thinking, processing languages, recognizing friends, reading books, learning, and reasoning. The diencephalon is composed of the limbic system, thalamus, hypothalamus, and pineal glands. The thalamus sorts and processes the prime sensory information of hearing, taste, touch and sight, but not smell. The hypothalamus is the control center of the autonomic motor system and it is a small almond-like structure found underneath the thalamus. Hypothalamus controls heart rate, body temperature, bladder contraction, and blood pressure, and maintains overall homeostasis. The pineal glands produce the melatonin hormone and involved in hormone regulation, together with the hypothalamus. The limbic system, also represented as the emotional brain, is accountable for basic emotional and survival behaviors.

The hindbrain consists of the brainstem, cerebellum, and medulla. The cerebellum, also known as the little brain, is the largest portion of the brain after the cerebrum. It coordinates voluntary movements, monitors the overall orientation of the body, and maintains balance and posture from the signals received from the inner ear. The brainstem act as the central network hub connecting the brain and the spinal cord. The midbrain is the topmost part of the brainstem and serves as the liaison between the forebrain and hindbrain.

As a part of the CNS, the spinal cord acts as a conduit between the brain and the other parts of the human body [50, 55, 56, 57]. The spinal cord contains a densely packed column of nerves that extend from the brainstem downwards to the spine. The spinal cord is divided into different segments based on the relationship established with different parts of the human body. The different segments are cervical, thoracic, lumbar, sacral and coccygeal. These five segments contain a total of thirty one pairs of spinal nerves that handle signals from different parts of the human body. For instance, the eight spinal nerve pairs (C1-C8) of cervical segment associated with the head, neck, arms, elbow, wrist and fingers in the human body. The functions of the spinal cord are to carry the signals from the receptors to the brain and to the motor nerves from the brain. The different segments of spinal cord are shown in Figure 16.

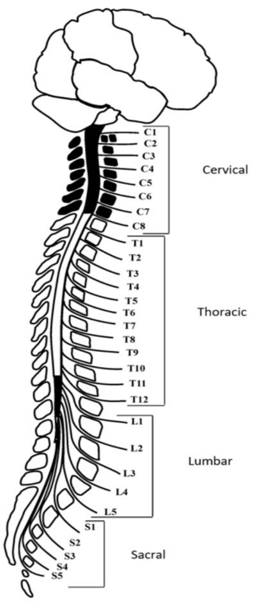


Figure 16. Spinal cord with segments

4.3.2 The peripheral nervous system

The peripheral nervous system (PNS) is made up of all HNS nerves other than CNS nerves. The PNS is composed of afferent nerves, also identified as sensory nerves, that sense events or activities, and efferent nerves, known as motor nerves, that cause action against sensory input. The PNS is further classified into two subdivisions: *somatic nervous system* (SNS) and *autonomic nervous system* (ANS) [49, 58, 50].

The SNS is associated with the controlling of voluntary muscle movement [49, 50, 59, 60, 61]. SNS's sensory nerves carry signals to the brain and spinal cord while motor nerves receive instructions and act accordingly. The SNS is responsible for all of our consciously induced body movements. Hence, it is also referred to as the voluntary nervous system. It also handles involuntary movement, known as reflexes, on certain conditions.

The ANS is also known as the vegetative nervous system or involuntary nervous system [49, 50, 58, 62, 63]. Unlike the SNS, the ANS is constantly active and regulates functions that cannot be consciously induced by humans. This regulation is executed through the signals received from the brain. It also sends information to the brain via afferent nerves. The ANS is further classified into two subclasses: the *sympathetic nervous system* and the *parasympathetic nervous system*. The former handles the responses related to physical and mental activity (e.g., increasing heart rate and preparing the body for self-defense), while the latter is accountable for resting responses, like salivation and metabolic processes. An overview of the human

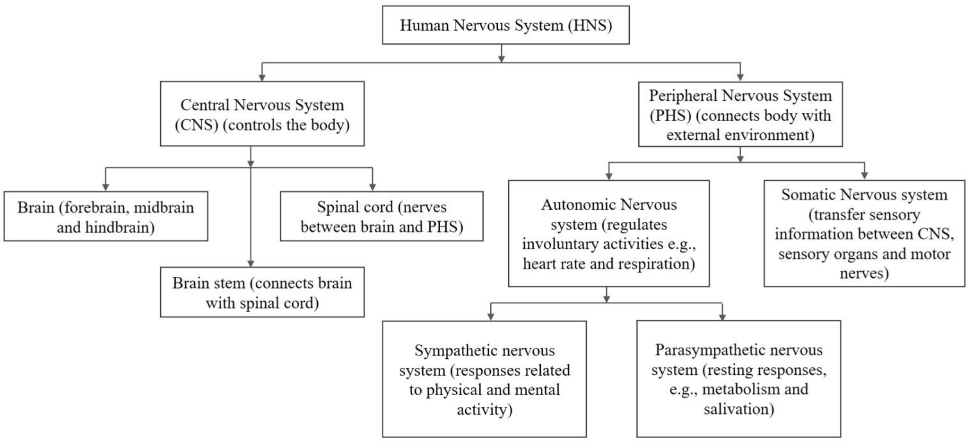

Human Nervous System (HNS)
Central Nervous System (CNS) (controls the body)
Peripheral Nervous System (PHS) (connects body with external environment)
Brain (forebrain, midbrain and hindbrain)
Spinal cord (nerves between brain and PHS)
Brain stem (connects brain with spinal cord)
Autonomic Nervous system (regulates involuntary activities e.g., heart rate and respiration)
Somatic Nervous system (transfer sensory information between CNS, sensory organs and motor nerves)
Sympathetic nervous system (responses related to physical and mental activity)
Parasympathetic nervous system (resting responses, e.g., metabolism and salivation)

Figure 17. Overview of the human nervous system

nervous system is shown in Figure 17.

4.3.3 Pain System

The somatovisceral system is made up of visceral and somatic nerves. The former handle the signals from soft internal organs such as the heart, lungs, reproductive organs, and circulatory and digestive systems. The latter handle signals from muscles, skin, and bones. The somatovisceral system has afferent nerves that carry signals to the CNS from receptors and send signals to effectors from the CNS through efferent nerves [58, 64]. For instance, when someone places a hand over a person's shoulder, the receptors attached to the skin generate and pass impulses to the brain using the somatic sensory system. The brain receives those impulses, takes decisive action against the impulses by generating response(s), and sends them as signals to the effectors. The effectors execute the signals that result in the movement of the head and eyes or changes in facial expressions.

The somatovisceral system originates from the dorsal horn, which is located in the spinal cord. The dorsal horn plays a crucial role in the handling of signals including pain and touch [64, 65]. A special pain receptor called the nociceptor handles the pain signals and forwards them to the CNS. The brain and/or spinal cord will make the decision based on the perceived intensity. Nociceptors are generally classified into A-fiber and C-fiber [64, 65, 66]. C-fiber nociceptors handle slow sensation signals and A-fibers mediate fast sensation signals.

The spinal cord will make decisions under certain critical circumstances. For example, when a part of the human body comes into contact with a hot surface, this results in instantaneous withdrawal followed by crying and shouting reactions. This instantaneous withdrawal response is decided by the dorsal horn part of the spinal cord without any involvement or interference from the brain. These kinds of responses are called reflexes. Other responses are decided by the brain depending on the perceived intensity of heat, damage caused during the encounter, and archived instances of similar previous encounters. In this example, the C-fiber nociceptor handles the prolonged burning sensation and the A-fiber nociceptor handles the instantaneous withdrawal and initial sharp burning pain, felt by the body surface that contacted the hot surface.

4.3.4 External Immunization

External immunization, also termed inoculation, is the process of helping the HIS to produce antibodies in order to fight invading foreign pathogens. The part of a pathogen that instigate the creation of antibodies is referred to as the antigen. Each antigen is unique, and it requires a specific antibody to neutralize the antigen along with the damaged cells. Several millions of antigens are capable of causing damage to the cells and preventing the normal functions of different systems in the human body. The HIS has the inherent ability to respond against any invading antigens into the human body. The duration of the HIS to react against an invading antigen depends on a few circumstances. For instance, HIS can take several days or even weeks to respond against an antigen during the first encounter in order to engineer a specific antibody for the neutralization of encountered antigen. Once HIS has successfully neutralized the antigen with an engineered antibody, the antibody will be stored to deploy against the same antigen in future encounters. Thus, HIS processes are inherently adaptive in nature. The adaptive immune system is extremely capable of reacting against millions of antigens in a highly specific way [44]. The process of HIS is shown in Figure 18.

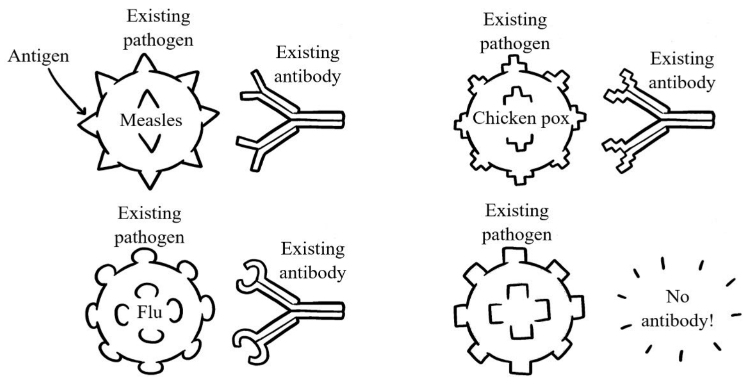


Figure 18. HIS process [67]

Inoculation is the process of familiarizing HIS with unknown antigens, which have not been encountered yet. During inoculation, the harmless inactive parts of the pathogen or outline of the antibody will be introduced into the human body in the form of injection or oral drops. Though the introduced parts are harmless, it will trigger the HIS to respond aggressively, which in turn produce the antibodies to suppress or eliminate the introduced pathogen. The produced antibody will be stored and then used against the same antigen when it invades the human body through infection [67, 68, 69]. The inoculation process reduces the reaction time of HIS when the antigen infects in real-time, since it already has the specific antibody. The inoculation process is also commonly known as vaccination.

4.3.5 HNS relationship with the HIS

The HIS plays a crucial role in defending the human body against invading foreign pathogens. The systematic functioning of the HNS is critical for the proper functioning of the human body, and any problem arising with HNS may result in serious consequences. For instance, any damage on the spinal cord leads to the non-functioning of parts connected with a particular segment; for instance, lumbar segment injury results in spasticity, bowel dysfunction, or even paralysis of the lower body.

Initially, the HIS and HNS were thought to be independent systems, but with the advancement of technology and medicine science, several studies carried out over the past five decades have established the existence of bi-directional communication between the two. The lymphoid organs, responsible for the production of antibody cells, are connected with ANS nerves, and hormones (catecholamine) released by the sympathetic nerves have impacts on immune cells. Furthermore, the immune system coordinates with the nervous system in regulating activities such as homeostasis [70, 71, 72, 73]. Even though further research must be performed to understand this relationship more clearly, with the current evidence, medical researchers concluded that the HNS influences HIS activity in reaction against pathogens, and the HIS influences the HNS in performing activities during rest.

5 Unique influential factors of the HNS for IT systems and security

The HNS is a complex system comprised of organs and a network of nerves. However, its operation is entirely structured. Every member of the HNS carry out their respective functions vividly and tacitly. The three major functions of the HNS that enticed us towards the HNS and in turn helped us identify the unique influential factors are discussed in this chapter.

5.1 Structural Organization

The structure of the HNS represents itself as an architecture that provides a framework comprised of components, their roles, interrelations, and communication among them. The brain is a multi-role, multi-tasking driving force that is capable of assuming several roles and processing several tasks simultaneously. For instance, it can be: a) a learner that performs learning and adaption, b) a decision-maker that creates and enforces decisions, c) a regulator that govern the initiation, control, and termination of activities. By assuming different roles, the brain protects the human body and ensures proper functioning by assuring minimum normal functions at any point in time. Unlike the brain, the spinal cord does not assume several roles but performs two critical functions: a) the transfer of signals between nerves and the brain and b) making decisions during emergency conditions in order to reduce damage or pacify internal activity. The entire operations of the brain rely on the signals passing through the spinal cord, and thus, it became an essential part of the HNS.

Nerve cells (i.e., neurons) are the foot soldiers of the HNS. They exist almost everywhere in the human body and establish very dense networks. Some estimates state that there are several billions of neurons ingrained in the human body. Without neurons, neither the brain nor the spinal cord can perform their functions properly. Hence, neurons should remain unscathed at all times, or if damaged, they should regenerate or repair themselves. When a nerve is damaged or cut, the axon will shrink and regenerate after a few days. The PNS has the intrinsic ability to repair and regenerate nerves from damage, and this process can be expedited using exercises and other strategies [192-195]. By contrast, the CNS's ability to repair and regenerate is limited. Any damage to the CNS can cause serious repercussions, and several studies have been conducted to encourage and improve the repair process of the CNS

[196-200].

5.2 Execution of communication flow and instructions

The communication flow of the HNS is complex and has unique aspects. The signals, detected by neurons, are converted into either chemical signals or electric impulses and transferred towards the spinal cord and then to the brain in order to elicit appropriate responses. Neurons are broadly classified into three classes: *afferent or sensory neurons*, *efferent or motor neurons*, and *interneurons*. Nevertheless, neurons are extensively diverse and hundreds of different types of neurons with unique message transfer abilities exist under the three classes. The establishment of a neuron network for communication involves several stages.

Neurons are created and sent to different regions. The term used to denote the formation of neurons is neurogenesis. Neurogenesis is a lifelong process that occurs during the development of the HNS at the initial stage and in adult humans [201, 202, 203]. After creation, neuron cells will travel to reach their destination, and this process is referred to as neuron migration [204-207]. One particular process in neuron migration that attracted our attention during the HNS study is cell molecule adhesion. The term, cell molecule adhesion may appear to be new, but the basic principle has been deployed in IT systems, though it is derived from another natural source. Under cell molecule adhesion process of neuron migration, neuron cells travel down the chemically marked path to reach the destination. This principle seems to be similar to ants that follow strong chemical traces to find a path to their destination (e.g., food source). Another interesting factor is that once a neuron cell reaches its destination, it will evolve (e.g., to be afferent or efferent) to fit into that exact location. The factors that promote neuron transformation are unclear. Thus, the HNS forms a network of nerves throughout the human body.

The execution of instructions is relatively simple as long as the established network remains intact. When execution requires the movement of certain parts of the body or system, motor neurons will be used. For instance, when someone touches our hand, a group of afferent neurons within the specific region sends signals to the brain via the spinal cord. Before delivering a response, the brain will derive answers to several questions, including: Which part of the hand was touched? How did the contact feel? Is the contacting object alive or not? Does the contact cause any harm to the contact surface? Is additional information (e.g., visual, audio) required to make a decision? and so on. If the contact is categorized as harmful, it will be handled as pain signals (see section 4.3.3 for pain signals). Finally, motor neurons execute the given decisions by exhibiting appropriate physical movements.

5.3 HNS relationship with other systems

The HNS maintains a relationship with other systems in the human body to cooperate and influence the functions of other systems individually or together in order to sustain the body's normal functions. The HNS-HIS relationship is discussed in section 4.3.5. The HNS also maintains a relationship with another system called the hormonal system. The hormonal system, also known as the endocrine system, is a hierarchical system composed of glands that are positioned in the different parts of the body that secrete and respond to hormones. This system is responsible for the maintenance of homeostasis and responds to environmental changes through hormones [208, 209]. The term endocrine refers to the response that produces hormones by the glands against specific stimuli [208].

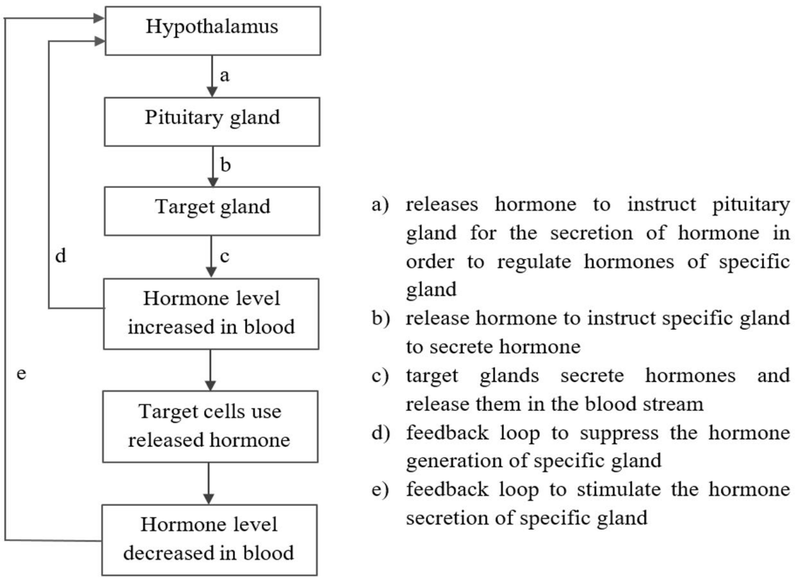

Hypothalamus
a
Pituitary gland
b
Target gland
c
d
Hormone level increased in blood
e
Target cells use released hormone
Hormone level decreased in blood
a) releases hormone to instruct pituitary gland for the secretion of hormone in order to regulate hormones of specific gland
b) release hormone to instruct specific gland to secrete hormone
c) target glands secrete hormones and release them in the blood stream
d) feedback loop to suppress the hormone generation of specific gland
e) feedback loop to stimulate the hormone secretion of specific gland

Figure 19. Overview of endocrine glands function and feedback loops

Hypothalamus, pituitary glands, thyroid, pancreas, adrenal, parathyroid, and gonads are the glands that make up the endocrine system. With the exclusion of the hypothalamus and pituitary glands, the others are collectively referred to as peripheral endocrine glands. The pituitary gland, located in the brain, is referred to as the master endocrine gland, as the hormone secreted by this gland regulates the hormone secretions of the rest. The hypothalamus of the brain regulates the pituitary gland. Hence, the hypothalamus holds the highest position in the hierarchy. The highest order organ sends a hormonal signal to other organs, which respond by discharging their own hormones into the bloodstream. With the help of feedback mechanisms, the hormones produced by the peripheral endocrine glands can be stimulated or repressed by higher-order glands. An overview of endocrine glands with feedback is illustrated in Figure 19. The hypothalamus connects the HNS with the endocrine system. The ANS of the HNS interacts with the endocrine system to secrete hormones to control the body's activities. For example, cardiovascular system [210] functions (e.g., heart rate) are influenced by the ANS and the hormones of the endocrine system [211, 212].

5.4 Unique features of the HNS for security implementation

The HNS has exhibited several unique features that are essential and greatly influence the functions of the IT systems, provided that they have been adapted properly. These unique features are as follows:

(a) *Self-learning*: HNS organs, by nature, have the capacity to learn from the received signals, and thus, they achieve awareness of internal and external conditions by themselves (i.e., self-awareness).

(b) *Hybrid architecture*: The brain is the CPU of the HNS; however, it also allows the spinal cord to respond during critical conditions without brain interference. Hence, it has a hybrid power structure as a decision-making power that is not centered in a single place.

(c) *Reactive nature*: The HNS and its combined activities with the other systems (e.g., HIS and endocrine system) are always reactive in nature (i.e., response based on the stimuli detected by the nerve cells).

(d) *Self-regulate*: Through decisions, the HNS initiates, maintains, and terminates activities either actively or passively with and without human consciousness. With this function, the HNS exhibits the trait of self-regulation.

(e) *Self-healing*: Though complete recovery depends on the intensity of the received damage or injury, the HNS has an inherent quality to repair itself, and this regeneration can be further stimulated through other external mechanisms. Hence, the HNS, by default, has a healing capacity.

(f) *Network awareness and migration*: The HNS has a clear understanding of the extent of the established network and the happenings over the network. In addition, it is also known to form the network and guide the neurons with different cues to place them in appropriate positions.

(g) *Different communication signals at various speeds*: The HNS employs electric impulses and chemical signals to notify stimuli and uses high speed and low-speed channels to communicate and enforce decisions during normal and critical conditions.

(h) *Receptors-effectors*: The ability to use a group of different kinds of neurons for information gathering and decision enforcement.

(i) *Co-operated regulation*: The HNS highlights its ability to work with other systems and regulate the activities of the human body and other systems in order to maintain the proper functioning of the human body.

In conclusion, the HNS is a part of the human body. It protects its own host from damages and regulates the functions of the host individually and collectively with other systems that exist in the host. Thus, the structure and activities of the HNS and its relationship with other systems offers an excellent base for the architecture of IT systems and security. The impact of incorporating HNS capabilities and features will make security an inalienable part of a system, guaranteeing minimum operational capability and security resilience even during critical conditions (e.g., threats).

6 Contributions

In this chapter, the contributions of the dissertation are explained in detail. The adaptation of the HNS is discussed in section 6.1 and implementation of the adapted HNS, through experiments, in IoT and mobile platforms are elaborated in sections 6.2 and 6.3 respectively. During experimentation, the newly designed system will undergo feasibility analyses and evaluation to determine implementation capability and fulfillment of defined objectives. We conducted a feasibility analysis in IoT and mobile device platforms for the development of prototypes and evaluations. The publications [218, 219, 221] represent the derivation of a new system architecture from HNS functionalities and refined architecture for the experiments. Prototype implementations and evaluations in terms of resource consumption and improvements in security for the experiments performed in IoT and Android are given in [220, 221].

6.1 Adaptation of the HNS

The adaptation of the HNS is the first contribution of the dissertation. Adaptation is a challenging task to perform due to the inclusion of organs and functions of the HNS. The brain performs several operations simultaneously. It is a learning engine, which learns from the signals sent by the network of nerves. The brain is the decision-maker and regulator of all activities performed by the human body, either actively or passively. The unique nature of this activity is that it can offer different kinds of responses to a specific event. The brain is also a repository engine which holds all information it encounters. The event retrieval capability of the brain depends on the frequency of events recollection from memory. The spinal cord is a high-speed highway for signals. It takes decisions during critical situations without brain consultation. While this may seem complex, it is an elegant process. The employment of different kinds of signals that are traveling at different speeds, employing effectors capable of producing different types of actions based on the given instructions, and the ability to defend and recover are some of the fascinating functions of the HNS. These functions provide an excellent base for designing systems ingrained with self-awareness and security, provided they are properly adapted.

The architectural design approach towards adaptation should reflect the HNS architecture, which is neither distributed nor centralized, and its cooperative nature. Apart from the functions of the brain, three areas should be addressed for adaptation.

(a) *Existence of different communication pathways*: The HNS has two different speed reconfigurable communication mediums: A fiber (high speed) and C fiber (low speed), for exchange of pain signals and appropriate responses.

(b) *Spinal cord*: It also acts as a decision-maker in critical situations to minimize the damage to the human body. The main point here is that the decisions made by the spinal cord do not have any interference from the brain neither contradict nor overwrite the brain's decision.

(c) *Receptors-effectors*: Though the existence of sensory and motor nerves is equally important in the adaptation process, effectors require additional attention, particularly at the operational level. The reason being that the brain and spinal cord use the same effector to produce different output through their instructions, for instance, changes in the amount of strength and force in the movement of body parts.

Architectural level changes in the system design are necessary in order to include the HNS components and address the three points listed above. Furthermore, the system should retain the HNS capabilities even after severe refinement based on the target field of implementation. This is explained in two cases. In case 1, computer security as a target area, desktops and laptops are capable of hosting entire functionalities and components of the HNS in stand-alone architecture. Under client-server architecture, desktops and laptops can be the host for nerve points and sections of the spinal cord to guarantee limited autonomy, the server host brain, and other functions of HNS. In case 2, IoT as target area, the case 1 approach cannot be applied, since RCDs are unable to host resource-intensive operations due to their limited resource capabilities.

The HNS-based security system should consist of individual subsystems for the brain and spinal cord along with the establishment of receptor-effector and dedicated communication. Operation-sharing is crucial for the system as it addresses functional preservation after refinement for specific target areas. The different subsystems should be capable of deploying in several devices or locations, and their actions should be coordinated through communication. The segregation of functions between the subsystems should be weighed on the requirements for execution against the offered requirements as it may vary between devices and IT application areas. Resource-intensive and fewer resource requirement tasks need to be identified and segregated. For instance, usage of the spinal cord subsystem alone helps to handle critical scenarios. The availability of sensory functions enables the spinal cord to independently perform learning within the deployed devices. The spinal cord can be deployed as a whole or can be divided further. This kind of operation-sharing mechanism will lead to the cooperation and coordination of several spinal cords with a single brain instance. The system architecture that comes as the result of operation sharing will be hybrid in nature as the decision making feature will be spread evenly.

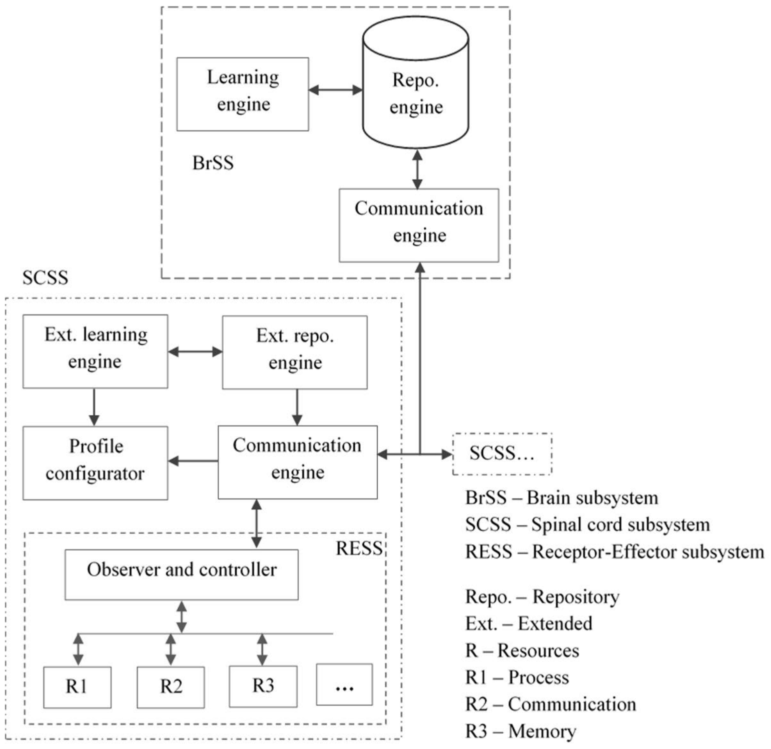


Figure 20. Overview of the HNS based system architecture

Communication similar to A-fiber and C-fiber nociceptor should be created in order to exchange signals at different speeds and situations. C-fiber communication should be used to carry signals during normal conditions (e.g., gather signals transmission), whereas A-fiber should be used for critical circumstances (e.g., notification of the abnormal incident). The establishment of receptors and effectors is also important, as these are used to send events and exert control over the resources. The creation of nerve links should be generic to enforce different-natured actions. To achieve this, crucial parameters for the deployed devices and their own execution should be identified, segregated, and grouped to create a mechanism profile, that can be managed automatically by the brain or spinal cord or manually. The configurations that must be regulated through the profile are memory management, evaluation process interval, type of communication and interval, category of decision, enforcement duration of decisions, decision communication duration, acceptable behaviors, and parameters for the learning process. This configuration shall be expanded further to include parameters of different target areas.

An overview of HNS-based system architecture is illustrated in Figure 20. It comprises three major subsystems, which in turn consist of several engines or modules to perform HNS functions. The brain subsystem (BrSS) is comprised of the learning engine, repository engine, and communication engine, and has the ability to manage several spinal cord subsystems (SCSSs). The repository engine gathers the signals that will be used by the learning engine from different SCSSs. The communication engine handles the exchange of signals and decisions between BrSSs and SCSSs.

The spinal cord subsystem consists of the receptor-effector subsystem (RESS), extended communication engine, learning and repository engine, and profile configurator. The extended learning and repository engines handle the learning and storage of signals within the deployed device at a limited scale. Based on the implementation area, the learning engine can engage in creating behavior patterns within the deployed devices or receiving patterns from the brain, and similarly, the repository engine can store the latest information and purge dated information. The communication engine handles external communication with the BrSS and other systems, and internal communication with the RESS. It employs both A-fiber and C-fiber for different situations. The RESS of the SCSS consists of an enforcer and controller that are responsible for the regulation of activities. For instance, the communication controller observes and regulates the activities taking place in the communication medium. The main advantage of HNS adaptation is the guaranteed threat resilience through hybrid architecture that ensures a rapid reaction against incidents and enhances system operations.

6.2 Enhancement of security in IoT using the proposed HNS architecture

The second contribution of this dissertation is the implementation of the proposed HNS-based security system architecture in IoT, particularly to handle internal attack cases through experiments. These experiments explored the possibilities of implementation feasibility together with the required changes in the proposed architecture. RCDs, the foot soldiers of IoT applications, engage in sensing and information exchange. Edge routers manage the foot soldiers and maintain the communication with external networks. Hence, the architecture should be refined to prevent a heavy toll on RCD resources.

6.2.1 Assumptions

Before proceeding further on the details of architecture refinement and implementation, assumptions that were made for this study are listed below:

(a) *Assumptions on edge router*: It has sufficient processing power to configure and maintain the network-wide cryptographic keys and storage capacity for storing contextual data and reported activities. The edge router is a fully secured unit and the probability of realizing unauthorized access is exceedingly unlikely.

(b) *Assumptions on IoT network*: The network is secured with AES-128 cryptography to prevent external attacks. Furthermore, the implementation assume that the network operations are secure at the beginning to learn the proper behaviour of nodes and their characteristics.

6.2.2 HNS-based architecture system design

The proposed HNS-based architecture is employed to design and implement the temperature-monitoring application with anomaly-based IDS. The objective of the application is to monitor and report the temperature in the designated areas and to protect the network from insider attacks. The IDS implementation enables nodes to monitor their child, 1-hop neighbors' behavior, and identify discrepancies through their network activities (e.g., packet size and data rate). All the gathered information is stored and analyzed locally within the respective nodes without any communication with other nodes. Communication with the parent node occurs only during notification of changes (inactivity, malicious activity, and new neighbor appearances) in the neighborhood. A new ICMPv6 based control messages, called Distress Progpagation Object (DPO), is used for reporting changes in the network, and the exchange of observed temperature information through UDP messages.

Under operation sharing, both nodes and edge routers should perform HNS functionalities, and thus, IDS require the compulsory participation of every node in the network. Since IoT devices are resource-constrained in nature, BrSSs and SCSSs should be simplified for edge routers and nodes, respectively. BrSS and SCSS retain their core components (e.g., controllers, profile, repository, and communication), but the names are changed to reflect the current implementation domain. The modified version of SCSS for RCDs specifically for IoT networks is presented in Figure 21.

The SCSS of the node is composed of three subsystems along with a profile and repository to detect an anomaly. These three subsystems carry out their functions independently without affecting others' operations. These are the *monitoring and grading subsystem* (MGSS), *isolation subsystem* (ISS), and *reporting subsystem* (RSS). The MGSS consists of information acquisition, and information analysis phases. In the information acquisition phase, the crucial details, like payload size, sender information, destination information, message type, and received timestamp, are extracted from the received packets, and appropriate data are derived from them. These derived data are compared against previously stored assessments of the same node in the information analysis phase. If any changes in pattern are identified, respective constants and flags will be changed. Finally, the assessment will be made on the constants, flags, and given profile configuration. The derived data, constants, and flags are stored temporarily in the node's data store.

Incident reporting and purging phases are part of the RSS, and their operations

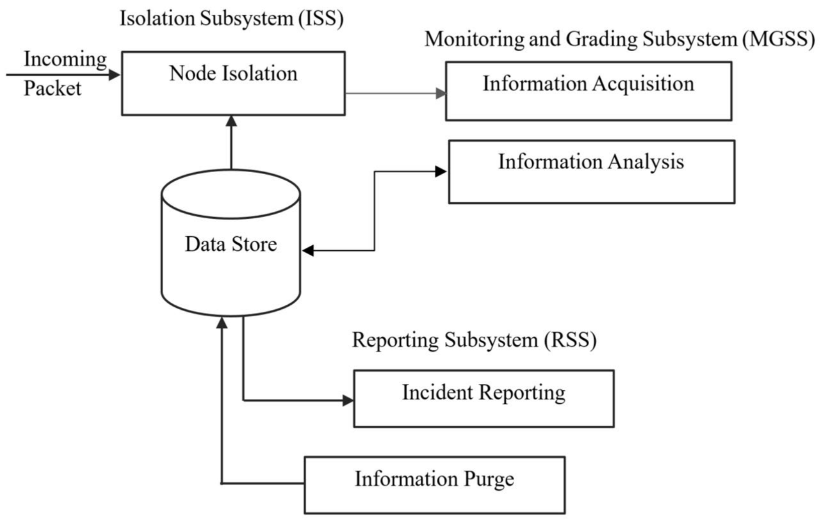


Figure 21. IoT specific refined version of spinal cord subsystem (SCSS) for RCDs

exclusively rely on the flags and constants manipulated by the information analysis phase. The anomaly-marked assessments are identified and sent to the parent node as incidents. A set of previously observed behavior will be purged on certain predefined intervals on a node-by-node basis due to the node's memory constraints. Node isolation of ISS is responsible for enforcing a communication ban on a node that fails to exhibit normal behavior. During packet inspection, it will verify the anomaly assessment with respective flags and constants for the corresponding node before allowing packets to the upper layers or discarding them.

Profile configuration consists of incident reporting intervals, subsystem execution, group grading, repository size, rating ranges, threshold window, and grading tolerance. It also specifies the duration of the communication ban. The profile configuration settings, optionally, can contain normal behaviors in the form of information related to the frequency and the size of data packets, control packets and employed cryptography techniques. Data and control packets information can be provided manually or the system can learn through observation, but the employed cryptographic technique should be given manually. Profile configuration can be defined on a node basis or for the whole network manually during programming or from edge-router. The primary objective of profile parameters like group grading and threshold window, is to reduce false positives, as anomaly detection systems generate false positives due to the lack of normal behavior awareness and variable network behavior. The rating range and grading tolerance parameters provide flexibility during the assessment. This flexibility offers additional time for a legitimate node, to correct the behavior, which may be raised due to genuine incidents like loss of packets or self-restart after an error in execution. Abnormal behavior due to power issues in legitimate nodes is also considered as an anomaly. If it is not self-correctable, say after the auto restart and the problem continues after the restart, it will be taken as an anomaly since the power issue is real and requires manual intervention to replace the energy source.

Edge routers receive notifications from nodes, process and correlate the received notifications, and make decisions based on them. The intermediary, subroutine, API, and data store modules form the BrSS of the HNS in edge router. The API module provides information services (e.g., reports on temperature and abnormal incidents) to the remote end-users. The subroutine performs the core functions (e.g., anomaly deduction and data correlation). It is responsible for the maintenance of a list of active nodes in its own network. The correlation function is executed on predefined intervals in order to verify consistency in the received data, notifications and network. If an inconsistency is reported by a non-parent node, the edge router will verify the reports of all neighboring nodes of the node that exhibited inconsistency. If the abnormal incident notification of a node comes from the parent node, the edge router makes an immediate decision, provided that the node's position information from all neighbors is satisfied. The SCSS in the node enforces a communication ban quickly and temporarily, whereas a BrSS will decide on making the ban permanent. Intermediaries act as communication engines and facilitate the exchange of information between the nodes and edge routers. They also manages the storage of information reported by nodes.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| RPLInstanceID | Version no.   |     Flag      |    length     |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|                                                               |
                            Payload
|                                                               |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Figure 22. DPO message format

A new set of ICMPv6 control messages called Distress Propagation Object (DPO) was developed with the primary purpose of communicating network changes, behaviors, and decisions between nodes and edge routers. The notable features of the DPO are unicast, delivery of multiple payload objects, and reactive communication (i.e., nodes will disseminate DPO messages only to notify the changes without the edge router request). Thus, it reduces traffic congestion and resource consumption in the network. DPO messages can carry neighbor information object (NIO) and observed neighbor information object (ONIO) as payloads.

The base format of the DPO is shown in Figure 22. The RPLInstanceID and version number field contain a network instance identifier and DODAG version, respectively. The flag field determines the type of payload, and the values of the flag

Table 7. Payload type and flag value

Payload type	**Flag**
Neighbor information	1
Inactive neighbor information	2
Anomaly in contextual data	3
Anomaly in control messages	4

field are given in Table 7. When a node receives DPO, it will check the RPL instance ID and version number to determine the message originated from the same network. If the match fails, it will discard the packet. Then it will check the flag field to deduce the message type. If the flag contains values other than predefined values (see table 7), the packet will be discarded. The NIO payload is used to notify neighboring nodes' information to the parent node. The ONIO payload is used by the nodes to propagate the inactive neighbors and anomaly information messages. The NIO and ONIO payload formats are illustrated Figures 23 and 24, respectively.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| RPLInstanceID |  Version no.  |     Flag      |    length     |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|     pFlag     |         MAC (64-bit)                          |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|                   Local scope IP (128-bit)                    |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Figure 23. Neighbor information object (NIO) payload format

A *neighbor information* message is used to propagate the NIO payload containing information pertaining to the neighbor nodes and the parent node. The MAC and LocalScope IP fields of NIO contain a MAC address and link-local IPv6 address, respectively. The pFlag field determines the node's parent status with respect to the sender node, and the payload length field holds the number of NIOs in the payload.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| RPLInstanceID |  Version no.  |     Flag      |    length     |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|                         MAC (64-bit)                          |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
|                   Local scope IP (128-bit)                    |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Figure 24. Observed neighbor information object (ONIO) payload format

An *inactive neighbor information* messages disseminate the ONIO payload to notify the parent node about the inactivity of a node that was an active neighbour and has not been communicating. The functions of MAC and LocalScope IP fields are similar to the NIO payload. This DPO message mandates acknowledgment from the root to confirm reception, as unauthorized removal and physical confiscation might be troublesome causes for network operations, apart from issues of mobility or power exhaustion.

An *anomaly information* message also uses the ONIO payload and carries the details of nodes that exhibited abnormal behavior. The flag field determines the type of anomaly. Unlike inactive neighbor information message, anomaly information message does not elicit any acknowledgment from the root. The reason behind this nature is due to the type 3 DPO message that specifically handles the anomaly in contextual data. These type 3 DPO messages can be disseminated from nodes to root and vice versa. The nodes will assess the whole packet based on the profile configuration, say, frequency and size of data packets, to determine an anomaly. They cannot open the received encrypted packets to compare the received data as the parent or the forwarding node function may be different from the sender node and may tax additional energy consumption. The identification of anomalies, at the data level, for the contextual data is executed at the edge-router during the data correlation process.

6.2.3 Implementation and evaluation

The temperature monitoring application is implemented using Contiki OS in both test-bed and simulation environments. The 6LoWPAN network is created using Contiki OS. The Cooja simulator is used to create a simulation environment. For test-bed, Z1 motes are used as nodes and the combination of PandaBoard and Z1 mote is used as an edge router. Two types of hardware are used for the edge router to fulfill two primary operations:

(a) *Communication with the external network*: PandaBoard acts as a Wi-Fi transceiver and Ethernet port to establish a connection with normal IP networks, but it does not have an IEEE 802.15.4 compliant transceiver for 6LoWPAN network communication.

(b) *Communication with the 6LoWPAN network*: Z1 mote has an IEEE 802.15.4 radio transceiver for 6LoWPAN network communication.

Hence, Z1 mote is connected to PandaBoard through a USB interface in order to enable PandaBoard as a proper gateway between 6LoWPAN and IP networks.

Since the IDS operation relies on the 6LoWPAN network, the 6LoWPAN communication stack is customized by introducing IDS subsystems. RPL protocol is modified to include DPO messages. It has codes to identify DIO, DIS, and DAO messages. Similarly, DPO message codes 0x04 and 0x05 are included in order for RPL to recognize them. In simulation, two different network topologies are used for evaluation – 20 nodes in the first network and 50 nodes in the second network, without root. The nodes are positioned in the network as such that each one will have at least three to five neighbors.

Three attack cases: selective forwarding, packet flooding, and clone attack, and two resource requirements: energy and memory, are studied during evaluation. In clone attack evaluations, identical copies of a legitimate node are introduced in the two different regions of the network as a malicious node. Neighboring nodes disseminated DPO with NIO payloads containing malicious node details to the root to inform them of the presence of a new neighbor, the newly introduced malicious node. The edge router's subroutine, a part of the BrSS, detected the inconsistency in the network as expected within 10-15 minutes after the malicious nodes' introduction due to the existence of three nodes having the same identity.

A malicious node used to discard the packets of a specific neighbor is programmed and introduced to the network for selective forwarding attack evaluation. As usual, the malicious node's neighbors propagated the NIO payload to the root. The edge router detected and raised alert of unreported inactivity inconsistency in the network due to the inexistence of communication with an active legitimate node in less than 10 minutes. For a packet flooding attack, four malicious nodes capable of transmitting 300 to 1200 packets per second are placed in different parts of the network. Neighboring nodes accepted packets from malicious nodes due to the unavailability of normal behavior, but they enforced a communication ban on malicious nodes in less than 25 seconds and dispatched an ONIO payload to the edge router in order to notify the system of abnormal changes in the network. In addition, the implemented temperature monitor application also displayed remarkable performance in memory and energy consumption. The occupied memory of 3315 bytes of ROM and 864 bytes of RAM is meagre for HNS functionalities, and when compared with similar approaches, it ranked among the lowest. In terms of energy, it introduced an additional 3.4 mJ consumption for every DPO message exchange.

Apart from the achieved results, based on our proposed HNS architecture, the implemented IDSs displayed notable features:

- Hybrid architecture: The concentration of decision making in resulting IDSs is spread to every device in the IoT due to the compulsory participation results in the hybrid architecture, which are similar to HNS architecture.

- Guaranteed threat resilience: Protection against threats by node without edge router involvement.

- Reactive nature security: Nodes actively assess the behaviors of their children and neighbors and react instinctively after the discovery of abnormal behaviors including nodes' disappearance due to mobility.

- Network fingerprinting: Knowledge of network changes and the positioning of nodes by the edge router without any location information hardware (e.g., geolocation).

- Low network congestion: the employment of a reactive type of communication and destination-oriented unicast messages rather than initiator-responder communication. Broadcast and multicast type of messages result in the few news messages in the network.
- Learning and grading: Helped to reduce false positives and prevent the loading of behavior information about RCDs.
- Self-regulation: Nodes engaged in regulating the network activities themselves during threat cases (e.g., packet flooding).
- Cross-layer design: HNS functions are spread between different layers of the communication stack to prevent execution of unnecessary operations (e.g., packet processing).

The BrSS refinement of the edge router allowed it to understand the changes in the network, behaviors of the nodes, decisions enforced by the nodes under attack conditions, location of a node at any given point of time, and regulate activities through personal decision enforcement. SCSS refinement specifically for RCDs proved that they can engage in functionalities similar to the spinal cord by taking decisions during critical times, even with little resources. In conclusion, the proposed HNS-based architecture is feasible to implement in a low-resource-available environment with negligible overhead. Additionally, the architecture offers security and self-awareness as a part of the system and works along with other security implementations.

6.3 Enhancement of mobile device operations through platform customization using the proposed HNS architecture

The third contribution of this dissertation is the implementation of our proposed HNS architecture in the mobile platform. The primary objectives of this implementation are to investigate the implementation possibilities of the proposed HNS architecture in mobile platforms and to enhance device operations and security. The aim of this contribution is to implement brain, spinal cord, receptor-effector, and voluntary control functionalities of the HNS architecture in the form of an access control system (ACS). The ACS regulates access requests to the resources in order to prevent unauthorized accesses, possible abuses, and exploitation. Since the ACS requires resource(s) to act, the identification of resources, exploitations and possible repercussions should be carried out before refining our HNS architecture specifically for ACS in the mobile platform. Furthermore, the limitations of the implemented security mechanisms in mobile platforms should be studied in order to identify and mitigate possible exploitation and abuse prior to the refinement of the proposed HNS architecture for ACS. The limitation study will also help to establish a relationship between implemented security mechanisms and HNS-based ACSs. Android is chosen for this study and will be customized for HNS architecture-based ACS implementation. HNS-based ACS is called CoDRA, which stands for **Co**ntext-based **D**ynamically **R**econfigurable **A**ccess control system for Android.

6.3.1 Limitation of implemented security mechanisms

Android relies on sandboxing and permission model mechanisms for restricting application activities. Sandboxing restricts the application activities within a confined space and the permission model enable applications to access the device functionalities in the form of permissions.

Open and unsecured resources: Several device resources are open and unsecured, as they seem to possess low or no risk through Android. However, they can be exploited by adversaries to perform user and device fingerprinting. Fingerprinting allows adversaries to gain knowledge about the behavior and designing unique threat models to target specific users or groups. There are several ways to perform fingerprinting, including a) analysis of USB connections, b) data from open and least/no secured resources, and c) list and categories of uninstalled and installed applications.

Though some of the resources are protected with permissions, they are not considered dangerous. Unregulated access to open passive sensors, least or non-secured resources may result in serious privacy violations and security breaches. For instance, unrestricted access to open passive sensor information results in the deduction of location and trails of user activities like jogging and walking [213-216].

Abuse of permission verification: Due to the introduction of runtime permission revocation, application crashes become unavoidable. In order to overcome this abnormal incident, Android introduced permission verification methods (e.g., checkCallingPermission() and checkSelfOrCallingPermission()) [217]. Android recommends using these methods to determine the access right status prior to resource access. The possibility of abusing these verification methods is very likely and can be exploited (e.g., permissions over claiming) to force users to grant access to their devices in order to use the application.

Classification and runtime permission revocation: Android classifies permissions under different categories (see subsection 3.2.2) based on individual assessments. For instance, READ_CONTACTS permission is categorized as dangerous as it exposes the list of stored contact information, whereas ACCESS_WIFI_STATE enables applications to gather network-related information such as access point details, security protocols, and IP addresses, and it is labeled as normal. With open sensor information and a combination of location service provider and network information, applications are able to accurately determine user location details and activities without location permission(s), which are necessary to obtain location information [213, 214, 215].

Runtime revocation of permissions is targeted to provide additional control to users in order to regulate application activities by either allowing or confining access to the resources at any time. Runtime revocation does not cover entire permissions, rather, it allows the management of dangerous permissions only. For instance, ACCESS_WIFI_STATE permission cannot be revoked under any circumstances since it is listed as normal. Application developers with ill intentions can overcome this limitation by abusing permission verification methods.

6.3.2 Assumptions

Assumptions similar to the second contribution have been made for this study. Application that access resources are considered a threat to the device functions and user privacy. They can employ several means to access open, least secure, and fully secured resources, and to perform fingerprinting. Consequently, fingerprinting is assumed as a serious threat. Developers are expected to follow the programming practices recommended for application development, but developers with malicious intentions do not follow the recommendations. The assumptions concerning Android and its users are as follows:

(a) The Android operating system, security mechanisms, system services, and kernel space are fully trusted.

(b) Users have minimum knowledge about the device functionalities and principles (e.g., sensors and security protocols of Wi-Fi networks).

(c) Devices will not be rooted by the user.

(d) Users will not grant root privileges to any applications.

6.3.3 Architecture

The proposed HNS architecture is employed for the design and implementation of CoDRA. The primary objective of this implementation is to actively and passively enforce voluntary regulation on application activities and to fulfill the limitations of the existing security mechanisms through cooperation. Android has layered architecture as shown in Figure 13. In a layered architecture, request calls travel in a top-down or bottom-up fashion. The origin of the calls usually depends on the placement of the resources and the components, which requires them to perform tasks using resources in the architecture. In Android, resource request calls generally originate from the top layer and are sent towards the middle layer, which in turn forwards them to the lower layer. This flow exposes possible places where the request calls can be regulated.

HNS architecture should be refined to fit into Android's layered architecture. Unlike the IoT devices, mobile devices have sufficient capacity in terms of energy, computation, and storage. Two factors that require attention during refinement are:

(a) Provisions for control and configuration: Since the target is a mobile device, users should be involved in the process. They require knowledge of the application activities and the available provisions for regulation and configuration of CoDRA.

(b) Communication and cooperation with other BrSS: As a scalable feature, it should coordinate with other mobile devices when the two are expected to work together.

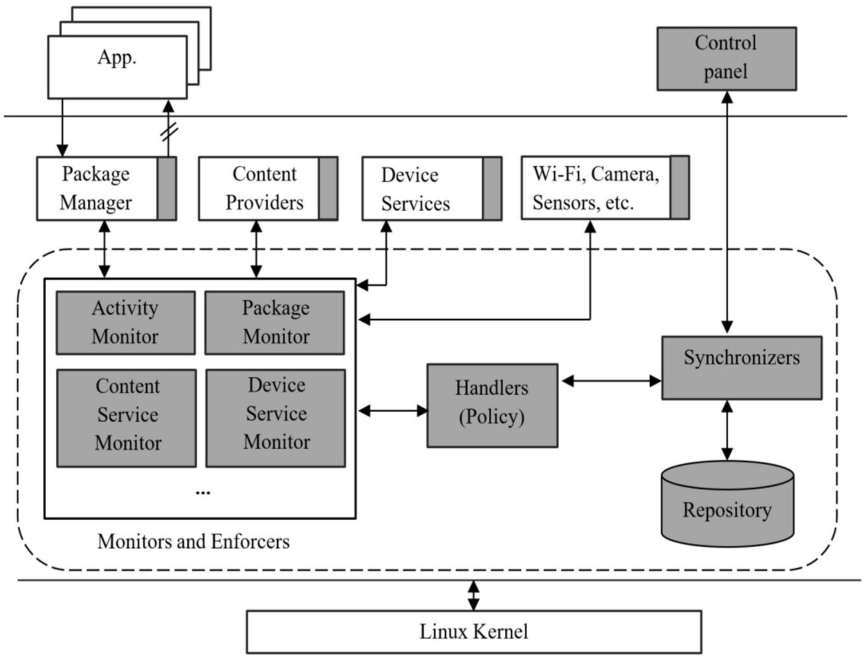


Figure 25. Customized Android architecture with proposed HNS architecture

The customized Android architecture, illustrated in Figure 25, is ingrained with our proposed HNS architecture. The refined HNS architecture for the mobile platform retains the functions of the HNS. The refinement introduced the third category of response, returning spurious information against request calls, in addition to the typical binary responses of allow and deny access. The BrSS consists of synchronizers, a repository, and a control panel where as the handlers, monitors, and enforces form the SCSS of CoDRA.

Monitors and enforces are the RESS of SCSS. They play a critical role in the

CoDRA by exposing the established observation and control links as policies. For example, a package monitor and enforcer expose observation and control links to regulate activities like the installation and uninstallation of applications. The sensors monitor and enforcer exposes links to regulate access to active and passive sensors (e.g., camera, accelerometer, and gyroscope). Through the established observation links, it can track the details and nature of request calls and execute responses through control links. Monitors and enforcers are designed in such a way that they will act on request calls after they pass through Android security mechanisms successfully. Handlers act as a conduit and aggregator for exposed links. They are responsible for communication responses to the control links, which respond to the calls by allowing, terminating, or returning fallacious data.

The control panel provides UI and exposes CoDRA's application activities and configuration. Synchronizers act as liaisons and decision-makers, that connect handlers with the control panel and repository. Synchronizers also enable the control panel to perform operations that are restricted by Android security mechanisms, as control panel operates from the application layer. Synchronizers accept request calls only from the control panel. On the SCSS, synchronizers receive intercepted request calls from handlers and return the decisions as responses. The repository acts as a data storage that holds the application's observed activities including its nature and the enforced decisions.

6.3.4 Policy design and enforcement classification

Policies are the links exposed by the RESS in order to regulate and exert control over the links. They are crucial for CoDRA, since it is an ACS based on the proposed HNS architecture. The efficiency, effectiveness, and usage of the ACS is determined by the policies. Three factors should be established prior to the policy design process:

(a) Identification of source: Origin, where the calls are initiated, and the features that help to establish the identity of the origin.

(b) Target: Generally, resources available in the devices are the target, which needs protection. The resources can be stored information or hardware used to generate information.

(c) Different ways to reach the target.

CoDRA policies are designed on the basis of context. The definition of context may vary as it depends on the system and it will affect the definition of the system (e.g., the RBAC). In ACS, primarily the environment and identification of an entity will be used as context. In CoDRA, context is defined based on the attributes or features of an entity rather than the environment or other factors.

The policies are associated with the operating environment of an entity in an

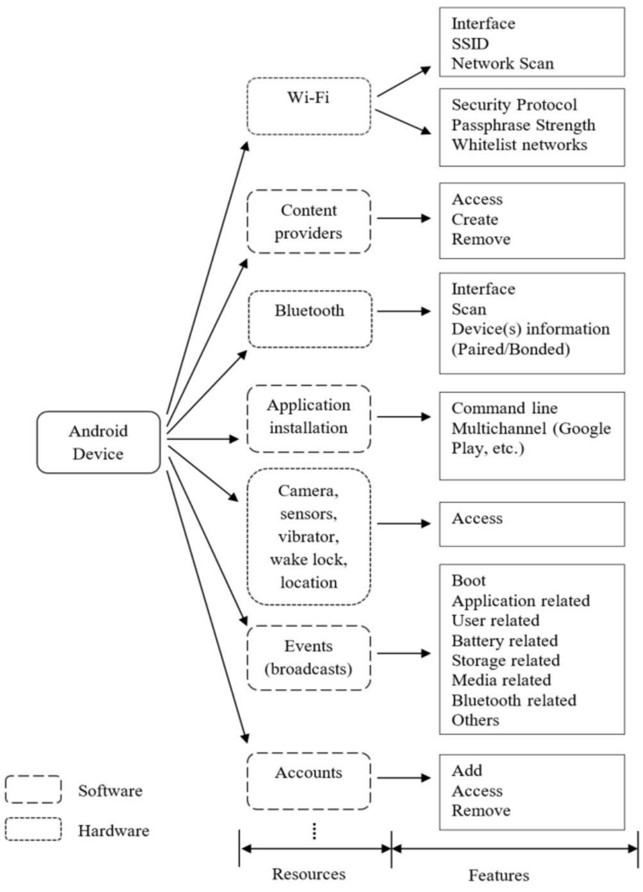


Figure 26. Classification of resources and their features in CoDRA

environment-based context, whereas in CoDRA, policies are associated with the attributes (i.e. features) and operations that can be performed or used by an application. The environment can also be used as an attribute in some cases, for example, modified time and GPS coordinates of resources. This will result in the establishment of highly refined policies for CoDRA. The information and hardware components of mobile devices are collectively referred to as resources. Key characteristics of resources are identified and classified as features. These features are used as the base for the creation of context-based highly granular policies. A partial list of resources and their related features is graphically illustrated in Figure 26.

Android activities are classified into two groups: system level and application level, and policies are congregated accordingly. System-level activities are operations that affect the entire device's functions. Application-level activities correspond to the operations performed by the application. For instance, a) accessing contacts and sensors are application-level activities and b) application installation and Wi-Fi association are system-level activities. Application-level policies are ineffectual in the regulation of system-level activities (e.g., the outcome of the application installation process determines the existence of an application).

The classification of enforcements is determined by the type and activity. System-level activities are regulated through system-wide enforcements, and application-wise enforcements control application-level activities. Enforcements are groups based on type: static and dynamic, in order to cover open and least-secured resources. The dynamic enforcement approach establishes policies during initial access request. It will verify the links' existence prior to creation. In the static approach, policies are established for secured resources only, and the verification of policy existence is unnecessary. Both approaches are used in CoDRA for policy establishment to regulate every resource.

6.3.5 Implementation and evaluation

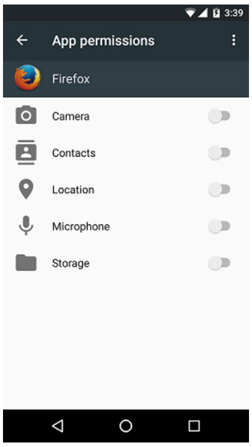
CoDRA implementation follows a layered architecture similar to Android architecture. The Android code base is customized and embedded with CoDRA components. The languages used for the implementation are C and Java. Control panels implemented using Java and core components are implemented using both C and Java languages. All CoDRA components, except control panel, are positioned in the Android's services and libraries layer. The monitors and enforcers intercept the request calls through the established links. Upon device boot, CoDRA establishes links to the resources and creates a repository for the management and storage of policies in the system partition. A common format for system-wide enforcements and a common structure for application-wise enforcement will be created, during the repository development, based on the established template (refer section 6.3.4). By default, the created policies status is set to active.

The application-wise enforcement template can be modified through the control panel. The CoDRA creates an individual policy structure for every application during installation from the application template. During installation, the process will be suspended and the information of the application: the UID, package name, and required permissions, are collected. Individual policy structures are created using the collected information, and the suspended process will be resumed after policy structure establishment.

The established observation links intercept the request call and forward the information: UID, USERID, package name, requested resource, and its feature, to handlers. The handlers, in cooperation with the synchronizers, determine the policy state and return the status decision. The control links return the status: originally requested information, spurious information, or deny access, to the application. System-wide restrictions handle activities such as network association, application installation, and user creation. Application-wise restrictions handle application-related activities through static and dynamic enforcements. CoDRA is fully integrated with the Android multiuser feature. With this integration, CoDRA allows different sets of policy structures and configurations for applications in different user accounts in Android mobile devices.

The control panel is implemented as a system application that comes as a part of the Android OS. Tab-based screen navigation UI is provided for the management of CoDRA. Four tabs are designed to access the configurations of system and user applications, device policies, and application policy templates. Control panel enable the users' to change the status (i.e., enable and disable) of the policies easily since each of them are implemented in the form of switch button (see Figure 27b). The test-bed environment is created using Nexus 5 and Nexus 9 devices, and the customized Android is installed in those devices for evaluation. More than 50 popular applications like Facebook, WhatsApp, Twitter and Snapchat, are used for evaluation purposes under real-time use cases. The evaluation of CoDRA is carried out in terms of policy granularity, enforcement of policies and operational overhead.

To determine the fine-grained nature of CoDRA policies, it is compared against the policies of the Android security mechanism, since one of the CoDRA's objectives is to fulfill the limitations of the implemented security mechanisms. The policy granularity of Firefox in Android and CoDRA is presented as screenshots in Figure 27. These two figures clearly highlight the difference in the degree of granularity and refinement of policies in CoDRA and others. Wi-Fi association is used as a case to assess the policy's refined nature in handling system-level activities. Wi-Fi association activity is regulated in CoDRA through the employed security protocols, passphrase strength and whitelisting networks. Wi-Fi policy configuration from the control panel is given in the screenshot in Figure 28a. Entire applications were


3:39
App permissions
Firefox
Camera
Contacts
Location
Microphone
Storage

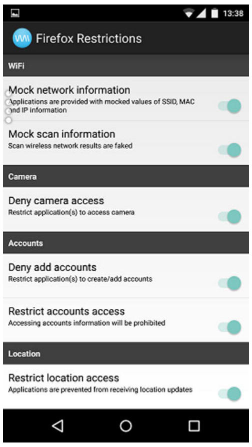
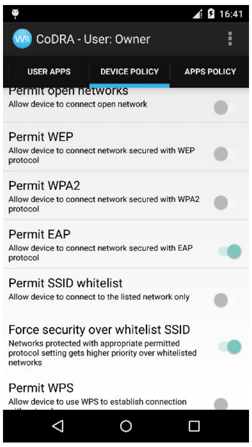

13:38
Firefox Restrictions
WiFi
Mock network information
Applications are provided with mocked values of SSID, MAC and IP information
Mock scan information
Scan wireless network results are faked
Camera
Deny camera access
Restrict application(s) to access camera
Accounts
Deny add accounts
Restrict application(s) to create/add accounts
Restrict accounts access
Accessing accounts information will be prohibited
Location
Restrict location access
Applications are prevented from receiving location updates

Figure 27. Firefox application policies: (a) Android settings and (b) CoDRA settings (partial list)


16:41

CoDRA - User: Owner

Permit open networks
Allow device to connect open network ○

Permit WEP
Allow device to connect network secured with WEP protocol ○

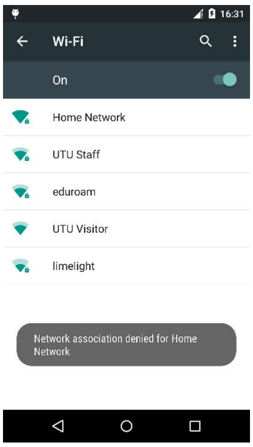
Permit WPA2
Allow device to connect network secured with WPA2 protocol ○

Permit EAP
Allow device to connect network secured with EAP protocol ●

Permit SSID whitelist
Allow device to connect to the listed network only ○

Force security over whitelist SSID
Networks protected with appropriate permitted protocol setting gets higher priority over whitelisted networks ●

Permit WPS
Allow device to use WPS to establish connection ○


16:31

Wi-Fi

On ●

Home Network
UTU Staff
eduroam
UTU Visitor
limelight

Network association denied for Home Network

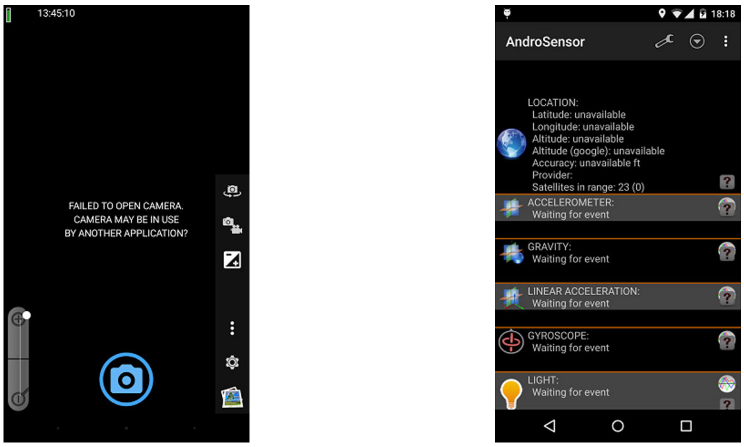
Figure 28. Wi-Fi association policies: (a) from control panel and (b) WPA2 association denial from Wi-Fi settings

executed with full restrictions and their resource request calls were captured, categorized, and stored under respective groups, such as Wi-Fi sensors, call logs, and Bluetooth. All possible broadcast events, like power, package, screen, and USB, were also logged. The Wi-Fi policy was set to restrict network association with all except the EAP and to remove the WPS. CoDRA successfully enforced the Wi-Fi policy and the network association screenshot is given in Figure 28b.

Application installation policies were set to prohibit installation from every source including Google Play. CoDRA prevented installation and returned its own error code (-125) denoting restricted activity. Since application restrictions were fully activated, resources should not be assessed by any applications. Applications that access active sensors (e.g, camera) and passive sensors (e.g., gyroscope and accelerometer), and also employ broadcasts were executed for verification. During execution, the sensors and broadcasts access request from these applications were returned with spurious response such that they won't either crash or stop the execution. The screenshots of Opencamera and Androsensor applications are presented in Figure 29.

An important event was observed during the evaluation: none of the applications crashed or failed to execute due to the enforcement of full restrictions, but crashes were detected when permissions were denied in the Android security mechanism. To calculate the additional execution overhead introduced by CoDRA, the same applications and test cases were executed in stock Android. CoDRA introduced an additional execution time of 1-20 milliseconds for the completion of every resource access request, which was meagre and unnoticed during usage. The additional execution time introduced by CoDRA for the execution of different resource access request is given in Table 8.

The memory requirement is directly related to the number of installed applica-


13:45:10

FAILED TO OPEN CAMERA.
CAMERA MAY BE IN USE
BY ANOTHER APPLICATION?

18:18

AndroSensor

LOCATION:
Latitude: unavailable
Longitude: unavailable
Altitude: unavailable
Altitude (google): unavailable
Accuracy: unavailable ft
Provider:
Satellites in range: 23 (0)

ACCELEROMETER:
Waiting for event

GRAVITY:
Waiting for event

LINEAR ACCELERATION:
Waiting for event

GYROSCOPE:
Waiting for event

LIGHT:
Waiting for event

Figure 29. Broadcast, active and passive sensor access denial: (a) camera denial for OpenCamera app and (b) passive sensors and broadcasts denial for Androsensor app

tions and user accounts in the device. At the initial stage, it consumed about 800 kB for device policies and the application-wise policy template. For every application, 5 kB to 9 kB were occupied, and further, an average of 5 kB were used for every user account. The memory requirement is exceedingly insignificant when compared against the CoDRA operation functionalities. In addition, mobile devices have at least 8 GB of internal storage, making the memory requirement of CoDRA infinitesimal.

Table 8. Execution overhead results (ms) for the average of 100 invocations

Resources	**Stock android**	**Customized android**	
		Allow	**Deny or return specious information**
Contacts	19.1654	28.5018	19.6989
Light sensor	7.1477	23.0150	19.3518
Location	10.5635	11.2263	11.1983
Wi-Fi information	1.3018	11.5942	9.5069
Camera	114.8567	129.5871	126.1546
Broadcast registration	22.1222	2.0396	22.0018

This experimentation in Android platform, therefore, proved the implementation capability of our proposed HNS architecture in mobile platforms and the fulfillment of its objectives, namely, the enhancement of device operations and security through cooperation with other security mechanisms.

7 Publication Summary

This chapter provides a summary of the original published research articles presented in the original publication section.

7.1 Publication I: Towards human bio-inspired defense mechanism for cyber security

In this publication, our inspiration was found in the human biological system. Our intention to apply the biological system in the field of cyber security to strengthen it from the design phase is presented clearly. The human nervous system, being one of the complex systems in the human body, inspired us greatly through its seamless flow of processes in executing operations. The functions and structures of HNS organs, especially the brain, spinal cord, events observation, respective response generation, and pain system, and its relationship with other internal systems of the human body are thoroughly studied. These structures and functions offer an excellent base for the system with in-grained self-awareness and guaranteed threat resilience. The basic functions, sensory, integration, and motor, are identified, along with the crucial phases, operation sharing and communication and receptor-effector establishment, which required caution during the adaptation of HNS functions into the field of cyber security. Through this study, a new security system architecture was derived from the HNS and was proposed for IT systems and security. The proposed architecture has a brain for decision making, learning, and storage, and a spinal cord for signal exchange and decision making during critical cases. Inclusion of similar components resulted in spreading the decision making capability reflecting the HNS. The primary objective of the proposed architecture is to employ security as an element (i.e., an integral part of the system itself).

Author Contribution: The ideation of concept, the study of the HNS, and the theory of HNS usage in cyber security were performed by Nanda Kumar Thanigaivelan, who is also the main contributor, author, and presenter of the publication at the conference , under the supervision of Dr. Ethiopia Nigussie, Dr. Seppo Virtanen, and Prof. Jouni Isoaho.

7.2 Publication II: Distributed internal anomaly detection system for the Internet-of-Things

This publication explores usage of the proposed HNS-based architecture from the previous publication in the IoT field. Here, the proposed architecture is refined, and a system design is developed from the refinement in order to create an anomaly-based intrusion detection system for IoT network security. The IDS is tasked with monitoring the 1-hop neighbor nodes' activities, reporting their abnormal behaviors, and networking changes due to factors like mobility and physical confiscation. The roles and functions are divided between the edge router and the nodes, and the placement of the IDS subsystem among those devices is clearly defined. Communication between the IoT devices is established using a new ICMPv6-based control message, which employs a reactive type of communication rather than the usual request-response communication pattern.

Author Contribution: The main idea was developed and presented in the publication by Nanda Kumar Thanigaivelan, who is also the main author, under the guidance of Dr. Ethiopia Nigussie, Dr. Seppo Virtanen and Prof. Jouni Isoaho.

7.3 Publication III: Hybrid internal anomaly detection system for the IoT: Reactive nodes with cross-layer operation

This publication presents the exploration of the implementation capability of our proposed architecture and the fulfillment of objectives in the IoT through the development and implementation of a temperature-monitoring application. The aim of this application is to monitor and report the temperature in the designated areas and protect its own network from internal attacks. The IDS, which will be part of the temperature-monitoring application, is implemented using the design proposed in Publication II. An edge router programmed to execute brain functions and nodes will utilize the spinal cord system. Nodes will learn and establish normal behavior through monitoring neighbor activities to track the presence of nodes in the neighborhood, report to the edge router and enforce a communication ban on nodes that failed to exhibit normal behavior. The edge router receives notifications on network activities, performs data correlation and threat deduction, and makes decisions. Contiki OS is used for the implementation of the application, and the 6LoWPAN protocol stack is customized for IDS implementation. The distress propagation object (DPO) is introduced to transmit the observed changes in the neighborhood to the edge router from the nodes through modification of the RPL protocol. For the evaluation, test-bed and simulation environments are created. The Cooja simulator created two different simulation environments with two network topologies comprised of 20 nodes and 50 nodes. For the test-bed, PandaBoard and the Z1 mote are combined to construct the edge router, and Z1 motes are used nodes. Clone attack, packet flooding, and selective forwarding attack cases are evaluated along with memory and energy consumption. Evaluation results showed that the proposed architecture reacts instinctively against threats and consumed a negligible amount of memory and energy compared to other related approaches. In addition, evaluation results also proved that nodes can make and enforce decisions to contain the threat.

Author Contribution: In this publication, the author, Nanda Kumar Thanigaivelan, had the primary role in the implementation of the proposed HNS-based system design in IoT, evaluating, and writing under the guidance of Dr. Ethiopia Nigussie, Dr. Seppo Virtanen and Prof. Jouni Isoaho.

7.4 Publication IV: Conceptual security system design for mobile platforms based on the human nervous system

A security system design specifically targeting mobile platforms based on the HNS architecture proposed in Publication I is presented in this publication. The proposed HNS architecture is refined for enhancing mobile security by addressing major changes in the segregation of operations between the brain and spinal cord, spawning receptor-effector, communication at different speeds, placing its components in the mobile platform architecture, and communicating with each other. Brain modules handle learning, incidents, and decision-making apart from the storage of incidents and decisions. The spinal cord modules manage the spawning process of receptor-effector pairs and the enforcement of decisions through them. A separate configurator offers a user interface for the management of brain and spinal cord modules. Challenges in the implementation of the proposed system design were studied. For instance, implementation through platform modification requires changes at the OS level, which in turn allows greater freedom for the brain and spinal cord modules, whereas as an application will limit the activities and resources that require protection against exploitation. This study highlights the advantages and disadvantages in implementing the proposed system design, either as an application or through platform modification.

Author Contribution: The author Nanda Kumar Thanigaivelan developed and presented the main idea in this publication. The author will be the main contributor and presenter of the paper at the conference under the supervision of Dr. Ethiopia Nigussie, Dr. Seppo Virtanen and Prof. Jouni Isoaho.

7.5 Publication V: CoDRA: A context-based dynamically reconfigurable access control system for Android

This publication presents the exploration of the implementation capability of our proposed architecture in Android platform through the implementation of a context-based dynamically reconfigurable access control system called CoDRA. CoDRA is developed on the foundation of the system design given in Publication IV through platform customization. The Android platform was chosen for this work due to the availability of source code. Android has a layered architecture in which the top layer hosts system and user applications, the middle layer consists of services, runtime, and native libraries, and the lower layer contains the kernel and device drivers. The middle layer is customized by introducing BrSS and SCSS components. SCSS establishes observation and control links through RESS on the device hardware and software resources (e.g., camera, sensors, call logs and messages, and device functions, in other words, application installation and Wi-Fi association through receptor-effector pairs in order to regulate the resource accessibility of applications). These latched receptor-effector pairs are exposed as policies. They are also used to monitor and send the activities of every application to the BrSS, which makes decisions accordingly. The decisions can be allowed, denied, or return random untrue information against the request for accessibility. A system application is developed to provide UI in order to configure and manage the functionalities of CoDRA and to understand the activities of the application through CoDRA's recorded log information. The test-bed environment is composed of devices, Nexus 9 and Nexus 5, are installed with the customized Android platform and for evaluation. The efficiency and effectiveness of the policies, decision enforcement overhead, and memory requirements are evaluated by analyzing over 50 popular applications. The evaluation results highlight the fulfillment of the objectives to controlling application activities and ensure application execution without necessary resource accessibility. Unlike similar approaches, the decisions are ternary in nature, which prevents the applications from forcefully granting resource accessibility. CoDRA introduced additional execution overhead in an average of 1.5 to 20 milliseconds depending on the resource. Furthermore, the evaluation results showed that CoDRA occupied 800 kB for device policies and templates, an additional 9 kB per application for policies, and approximately 5 kB for additional user profiles. Memory requirements are meagre compared to similar approaches, as mobile devices usually use a minimum of 8 GB for data storage.

Author Contribution: In this publication, the author, Nanda Kumar Thanigaivelan had the primary role in evaluating and writing the paper on the implementation of the proposed HNS-based system design in the Android platform under the guidance of Dr. Ethiopia Nigussie, Dr. Seppo Virtanen, and Prof. Jouni Isoaho.

8 Conclusion

In this dissertation, we identified and provided solutions for problems (lacking and restrictions) in the current system design approaches, which require changes to include security as a part of a system in order to improve the overall efficiency of the system functions and guarantee threat resilience. The IT system plays a very important role in day-to-day activities, and the transition towards the IoT brings IT even closer to human life. Any failure or security incident may result in privacy violation, monetary loss, or physical harm. Hence, security becomes a crucial component for IT systems. Unfortunately, these systems have been treated as tools rather than as parts or components of a system.

We presented a solution to this problem through the employment of a novel biological approach. The functions of the human nervous system in actively and passively protecting the host inspired us to use the same approach to solve the problem. The human nervous system has different organs with distinct purposes and an extensive network of nerves that sense any changes in the environment, both internally and externally. The HNS reacts against these changes to regulate and reduce the impact on the host individually or collectively. The architecture of the HNS itself presents a solid foundation, and has several influential features, like learning, a hybrid model, cooperation, a reactive nature, and healing, which will enable the derivations to acquire the same features. In this work, an adaptation of HNS architecture for IT systems and security was carried out. The derived HNS-based architecture was then evaluated using experiments, including the prototype, to prove the fulfillment of motivations and objectives defined in section 1.1.

The adaptation of HNS functions created a new architecture for IT systems and security. The structural organization, organs, and functions of HNS were studied thoroughly in order to become familiarized with all HNS operations, and critical areas were identified for the adaptation process. During the adaptation process, individual system components were created to represent similar HNS functions. These components were grouped to form individual subsystems. The interrelation between the subsystems and the components was created to imitate the HNS structure, thus resulting in the formation of system architecture embedded with security.

In the first part of exploration phase, HNS architecture was employed in the IoT domain and was refined with respect to IoT devices due to their resource-constrained nature. The functions of the BrSS and SCSS were segregated based on resource intensiveness, and were divided between edge routers and RCDs. Thus, the resulting architecture implicates participation from every node in the network. A temperature-monitoring application with an anomaly-based IDS was implemented in Contiki OS. The application monitors the temperature variations in the designated area and reports the findings to the edge router. IDS was responsible for the operations of the application, including communication activities and activities regulation, along with reporting to edge-router. Test-bed network (Z1 motes and pandaboard+Z1 as edge-router) and two simulation environments in Cooja were created for evaluation. Three threat cases: selective forwarding, packet flooding, and Sybil, were employed for the performance assessment. From the results of the test-bed and simulation environments, the proposed HNS-based implementation demonstrated: a) reactive communication and decisions, b) a hybrid architecture that allows decision-making in every node, c) learning and grading the activities of neighbors, d) network awareness, and e) instinctive rapid responses against abnormal behaviors. In addition, the results highlighted the low memory and energy consumption when compared to similar proposed research solutions. These demonstrated outcomes satisfied the defined objectives for the HNS architecture-based implementation in the IoT domain.

For the second exploration phase, a mobile platform was selected, and Android OS was chosen for the prototype design and development. The adapted HNS architecture was tailored to suit Android OS customization. An IDS was designed and implemented based on the refined HNS architecture by enabling the RESS of SCSS to establish and expose observation and monitor links, which were used as nerve points for sensing changes and regulating events on every resource request call originating from the Android applications. These links were exposed as policies and employed to regulate system-wide and application-wise activities. The BrSS makes decisions and stores decisions along with the observed activities. A control panel was implemented to provide an interface to the HNS-based IDS. It was implemented using C and Java, and Nexus 5 and Nexus 9 devices were used to form test-bed environments for the evaluation. More than 50 popular applications were evaluated under real-time usage. Evaluation results confirmed that the customized Android acquired enhanced operational capability and security. Its augmented capability in restricting applications from accessing resources by tracing the origin and different accessibility means without crashing the applications was demonstrated by the evaluation. This also proved that it fulfilled most of the problems addressed in section 3.3, particularly cooperation among security mechanisms. The customized Android consumed meagre amounts of memory and execution overhead for the operations of HNS-based IDS.

Through our work, we have established that system security will increase overall system efficiency and guarantee threat resilience at any given time. Furthermore, we also proved that the proposed HNS-based architecture can be refined as per the requirements of the field of deployment and its implementation capability through our experiments in two different domains. In conclusion, our proposed HNS-based system architecture is a promising solution for ingraining security as a part of the system and enhancing overall operational capability through cooperation among security mechanisms, hybrid architectural models, operational environment awareness, reactive natured communication, and regulation of activities.

8.1 Challenges and limitations

The human nervous system presents itself as a complex, yet eloquent, living functioning system having a relationship with other systems like the endocrine system and the human immune system to influence and control the operations of the entire human body both physically and mentally. Our entire journey is filled with interesting challenges starting from adaptation to implementation.

The primary reasons are HNS network establishment, execution of operations and relationship with other systems. Some of the HNS functions are still a mystery. For example, during the generation of nerve cells, they are all same and guided to their destination using various methods. Upon reaching the destination, they evolve themselves into different neuron cells capable of performing tasks distinctively, like sensing, executing, influencing and repairing, to fit perfectly as per the requirements of the destination. The causes behind this autonomous transition and perfect positioning are still unclear even after the tremendous advancement in the medical field. A fascinating and unique trait is that HNS influences its' own operations by influencing other systems. For instance, HNS bonded with the endocrine system that generates and releases hormones into the bloodstream to increase its ability during stress conditions, say, threatened, and pacify its own excited state. Adapting all functions require severe changes in the existing IT system development and security implementation. It can be theoretically executable but in practice, IT has several domains based on the usage and area of application. Each domain has its own set of devices that may foist certain constraints. Limiting changes in IT to narrow it down to a few specific IT domains posed a great challenge as it directly related to the selection or refinement of HNS functionalities for the adaptation process of individual domains.

IoT environment has devices, which have the least energy, memory and computation power, and are capable of operating unattended for a few years. The challenge we faced during the implementation is to retain the HNS functions against constrained resource availability. This challenge is solved through sharing of operations between nodes and edge-routers. Operation sharing solved challenges such as dynamic memory management, individual decision making and false-positives reduction. In our mobile platform experimentation, we faced a different kind of challenge in the form of implementation: platform customization or an application. Each has unique advantages over the other but we chose the former than using non-recommended development practices to establish an HNS network by dynamically hooking the request calls and compromising platform stability. This action restricted the dissemination of the research outcome to the public domain as it mandated skills for usage.

8.2 Future work

The proposed HNS-based system architecture will be extended to study its benefits in the other domains, particularly in mobile application development and OS. Since mobile applications enable users to access information and services through mobile devices from anywhere in almost every IT domain, the immediate emphasis is on the exploration of our HNS-based architecture in mobile application development. Our primary focus is to analyze and improve the security of the mobile device through applications based on the principles of HNS-based architecture by addressing the features described in section 5.4. Unlike the experimental studies presented in the sections 6.2 and 6.3, one of the goals for this exploration is to produce tangible outcomes, which can be publicly disseminated and used in real-time rather than confining them to research experiments. Our next focus will be on the employment of the proposed HNS architecture in the OS to study possible benefits and improvements in terms of increasing security and overall operational competency.

List of References

[1] Crypto Museum. The Thing. [Online] Available: https://www.cryptomuseum.com/covert/bugs/thing/index.htm [accessed 16.06.2021]

[2] Cryptome. Resonant Cavity Microphone. [Online] Available: https://cryptome.org/2014/12/ru-cavity-microphone.pdf [accessed 16.06.2021]

[3] Crypto Museum. Easy Chair. [Online] Available: https://www.cryptomuseum.com/covert/bugs/ec/index.htm [accessed 16.06.2021]

[4] Crypto Museum. Rocking Chair. [Online] Available: https://www.cryptomuseum.com/covert/bugs/ec/rc/index.htm [accessed 16.06.2021]

[5] P. Y. Papalambros. Design Science: Why, What and How. Design Science, vol. 1, p. e1, 2015.

[6] Android. What is Android - Android. [Online] Available: https://www.android.com/what-is-android/ [accessed 16.06.2021]

[7] Linux Kernel Organization Inc. The Linux Kernel Archives. [Online] Available: https://www.kernel.org/ [accessed 16.06.2021]

[8] Embedded Linux Wiki. Panda Board - eLinux.org. [Online] Available: https://elinux.org/PandaBoard [accessed 16.06.2021]

[9] Zolertia Wiki. Z1 – Zolertia. [Online] Available: http://wiki.zolertia.com/wiki/index.php/Z1 [accessed 16.06.2021]

[10] Contiki. GitHub - contiki-os/contiki: The official git repository for Contiki, the open source OS for the Internet of Things. [Online] Available: https://github.com/contiki-os/contiki [accessed 16.06.2021]

[11] Contiki NG. Tutorial: Running Contiki-NG in Cooja - contiki-ng/contiki-ng Wiki - GitHub. [Online] Available: https://github.com/contiki-ng/contiki-ng/wiki/Tutorial:-Running-Contiki-NG-in-Cooja [accessed 16.06.2021]

[12] I. Somerville. Software Engineering, Eighth Edition. Pearson Education Limited, 2007.

[13] R. Ross, V. Pillitteri, R. Graubart, D. Bodeau and R. Mcquaid. Developing Cyber Resilient Systems: A Systems Security Engineering Approach. National Institute of Standards and Technology, Special Publication 800-160, vol. 2, November 2019.

[14] pfSense. pfSense - World's most trused open source firewall. [Online] Available: https://www.pfsense.org/ [accessed 19.01.2022]

[15] OPNsense. OPNsense is true open source security platform and more. [Online] Available: https://opnsense.org/ [accessed 19.01.2022]

[16] Netfilter. Netfilter/iptables project homepage. [Online] Available: https://www.netfilter.org/ [accessed 19.01.2022]

[17] Cisco. Cisco Adaptive Security Appliance (ASA) Software. [Online] Available: https://www.cisco.com/c/en/us/products/security/adaptive-security-appliance-asa-software/index.html [accessed 19.01.2022]

[18] Cisco. Cisco Firepower 4100 Series - NGFW Appliances - Cisco. [Online] Available: https://www.cisco.com/c/en/us/products/security/firepower-4100-series/index.html [accessed 19.01.2022]

[19] Fortinet. Next Generate Firewall (NGFW). [Online] Available: https://www.fortinet.com/products/next-generation-firewall [accessed 19.01.2022]

[20] W. James. The Principles of Psychology, vol. 1, chapter 10. Dover Publications, 1950.

[21] A. Tagini and A. Raffone. The ‘I’ and the ‘Me’ in self-referential awareness: a neurocognitive hypothesis. Cognitive processing, vol. 11, pp. 9-20, 2010.

[22] A. Morin. Levels of consciousness and self-awareness: A comparison and integration of various neurocognitive views. Consciousness and Cognition, vol. 15, no. 2, pp. 358-371, 2006.

[23] J. O. Kephart and D. M. Chess. The vision of autonomic computing. Computer, vol. 36, no. 1, pp. 41-50, 2003, IEEE Computer Society.

[24] R. Sterritt. Autonomic computing. Innovations in System Software Engineering 1, pp.79-88, 2005.

[25] E. Amir, M. L. Anderson and V. K. Chaudhri. Report on darpa workshop on self-aware computer systems. UIUC Computer Science, Technical Report, 2007. [Online] Available: http://hdl.handle.net/2142/11289

[26] J. Isoaho, S. Virtanen and J. Plosila. Current Challenges in Embedded Communication Systems. International Journal of Embedded and Real-Time Communication Systems (IJERTCS), vol. 1, no. 1, pp. 1-21, 2010.

[27] Safe Wings Ottawa – Bird Collision Research, Prevention and Rescue. A look at bird-window collision statistics collected by Safe Wings in 2019. [Online] Available: https://safewings.ca/a-look-at-bird-window-collision-stats-collected-by-safe-wings-in-2019 [accessed 10.12.2021]

[28] National Audubon Society. Building collisions kill hundreds millions of birds per year. [Online] Available: https://www.audubon.org/news/building-collisions-kill-hundreds-millions-birds-year [accessed 10.12.2021]

[29] U.S Fish and Wildlife Service. Migratory Bird Mortality. [Online] Available: https://www.fws.gov/birds/bird-enthusiasts/threats-to-birds.php [accessed 10.12.2021]

[30] The Spruce. Prevent birds from hitting windows. [Online] Available: https://www.thespruce.com/prevent-bird-window-collisions-386497 [accessed 10.12.2021]

[31] Ask nature. Web decorations warn birds. [Online] Available: https://asknature.org/strategy/web-decorations-warn-birds [accessed 10.12.2021]

[32] National Audubon Society. Reducing collisions with glass. [Online] Available: https://www.audubon.org/news/reducing-collisions-glass [accessed 10.12.2021]

[33] University of Washington – Conservation Magazine. Windows inspired by spider webs. [Online] Available: https://www.conservationmagazine.org/2012/09/windows-inspired-by-spider-webs-2 [accessed 10.12.2021]

[34] Ask nature. Bird-friendly glass inspired by spider webs. [Online] Available: https://asknature.org/innovation/bird-friendly-glass-inspired-by-spider-webs [accessed 10.12.2021]

[35] C. L Craig and G. D. Bernard. Insect Attraction to Ultraviolet-Reflecting Spider Webs and Web Decorations. Ecology, vol. 71, no. 2, pp. 616-623, Ecological Society of America, 1990.

[36] S. Zschokke. Ultraviolet Reflectance of Spiders and Their Webs. The Journal of Arachnology, vol. 30, no. 2, pp. 246-254, American Arachnological Society, 2002

[37] G. Beni and J. Wang. Swarm Intelligence in Cellular Robotic Systems. In Proceedings of the NATO Advanced Workshop on Robots and Biological Systems, 1989.

[38] R. Beckers, J. L. Deneubourg and S. Goss. Trails and U-turns in the selection of a path by the ant Lasius niger. Journal of Theoretical Biology, vol. 159, no. 4, pp. 397-415, 1992.

[39] S. Goss, S. Aron, J. L. Deneubourg and J. M. Pasteels. Self-organized shortcuts in the Argentine ant, Naturwissenschaften 76, pp. 579-581, 1989.

[40] M. Dorigo and L. M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 53-66, April 1997.

[41] D. Merkle, M. Middendorf and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. IEEE Transactions on Evolutionary Computation, vol. 6, no. 4, pp. 333-346, August 2002.

[42] A. Colorni, M. Dorigo, V. Maniezzo and M. Trubian. Ant system for job-shop scheduling. Belgian Journal of Operations Research Statistics and Computer Science, vol. 34, no. 1, pp. 39-53, 1994.

[43] P. Brucker, A. Drexl, R. Möhring, K. Neumann and E. Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research, Vol. 112, no. 1, pp. 3-41, 1999.

[44] B. Alberts. Molecular Biology of the cell, 6th Edition. Garland Science, pp.1297-1342, 2015.

[45] F. S. Paula, M. A. Reis, D. A. M. Fernandes and P. L. Geus. Adenoids: a hybrid IDS based on the immune system. In Proceedings of the 9th International Conference on Neural Information Processing (ICONIP), vol. 3, pp. 1479-1484, 2002.

[46] J. Greensmith, J. Twycross and U. Aickelin. Dendritic Cells for Anomaly Detection. IEEE International Conference on Evolutionary Computation, pp. 664-671, 2006.

[47] L. Hong. Immune Mechanism Based Intrusion Detection Systems. International Conference on Networks Security, Wireless Communications and Trusted Computing, pp. 568-571, 2009.

[48] SEER Training Modules - National Cancer Institute. Nervous System. [Online] Available: https://training.seer.cancer.gov/anatomy/nervous [accessed 16.12.2020]

[49] Institute for Quality and Efficiency in Health Care. How does the nervous system work? [Online] Available: https://www.ncbi.nlm.nih.gov/books/NBK279390/ [accessed 15.12.2020]

[50] P. Rea. Introduction to the Nervous System. Editor(s): Paul Rea, Clinical Anatomy of the Cranial Nerves, Academic Press, pp. xv-xxix, 2014.

[51] G. Knott and Z. Molnar. Cells of the Nervous System. In eLS, John Wiley & Sons Ltd, 2001.

[52] S. Ackerman. Discovering the Brain. National Academies Press (US), Washington (DC), 1992.

[53] Queensland Brain Institute. Brain Anatomy. [Online] Available: https://qbi.uq.edu.au/brain/brain-anatomy [accessed 15.12.2020]

[54] National Institute of Neurological Disorders and Stroke. Brain Basics: Know Your Brain. [Online] Available: https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Know-Your-Brain [accessed 15.12.2020]

[55] A. Nogradi and G. Vrbova. Anatomy and Physiology of the Spinal Cord. In Madame Curie Bioscience Database [Internet], Landes Bioscience, 2000-2013.

[56] Queensland Brain Institute. The Spinal Cord. [Online] Available: https://qbi.uq.edu.au/brain/brain-anatomy/spinal-cord [accessed 15.12.2020]

[57] D. Purves, G. J. Augustine and D. Fitzpatrick. Neuroscience, Second edition. Sinauer Associates, Sunderland (MA), 2001.

[58] J. Chawla. Peripheral nervous system anatomy. [Online]. Available: http://emedicine.medscape.com/article/1948687-overview [accessed: 15.12.2020]

[59] M. A. Akinrodoye and F. Lui. Neuroanatomy, Somatic Nervous System. In StatPearls [Internet], StatPearls Publishing, Treasure Island (FL).

[60] J. Cuevas. The Somatic Nervous System. xPharm: The Comprehensive Pharmacology Reference, Elsevier, pp. 1-13, 2007.

[61] J. H. Kaas. Chapter 30 - Somatosensory System. The Human Nervous System, Third Edition, Academic Press, pp. 1074-1109, 2012.

[62] S. E. Borst, S. Goswami, D. T. Lowenthal and D. Newell. Autonomic Nervous System, Encyclopedia of Gerontology, Second Edition, Elsevier, pp. 129-135, 2007.

[63] Queensland Brain Institute. Autonomic Nervous System. [Online] Available: https://qbi.uq.edu.au/brain/brain-anatomy/peripheral-nervous-system/autonomic-nervous-system [accessed 15.12.2020]

[64] K. N. Westlund and W. D. Willis. Chapter 32 - Pain System, Editor(s): Jürgen K. Mai, George Paxinos. The Human Nervous System, Third Edition, Academic Press, pp. 1144-1186, 2012.

[65] A. G. Brown. Review article the dorsal horn of the spinal cord. Quarterly Journal of Experimental Physiology, vol. 67, no. 2, pp. 193–212, 1982.

[66] C. L. Stucky, M. S. Gold, and X. Zhang. Mechanisms of pain. In Proceedings of the National Academy of Sciences, vol. 98, no. 21, pp. 11845-11846, 2001.

[67] World Health Organization (WHO). How do vaccines work? [Online] Available: https://www.who.int/news-room/feature-stories/detail/how-do-vaccines-work [accessed: 16.12.2020]

[68] Centers for Disease Control and Prevention (CDC). Understanding How Vaccines Work. [Online] Available: https://www.cdc.gov/vaccines/hcp/conversations/understanding-vacc-work.html [accessed: 16.12.2020]

[69] British Society of Immunology (BSI). How vaccines work. [Online] Available: https://www.immunology.org/celebrate-vaccines/public-engagement/guide-childhood-vaccinations/how-vaccines-work [accessed: 16.12.2020]

[70] M. J. Kenney and C. K. Ganta. Autonomic nervous system and immune system interactions. Comprehensive Physiology, vol. 4, no. 3, pp. 1177-1200, 2014.

[71] T. Ziemssen and S. Kern. Psychoneuroimmunology – Cross-talk between the immune and nervous systems. Journal of Neurology, vol. 254, II8-II11, 2007.

[72] R. Dantzer. Neuroimmune Interactions: From the Brain to the Immune System and Vice Versa. Physiological reviews, vol. 98, no. 1, pp. 477-504, 2018.

[73] D. Lorton, C. L. Lubahn, C. Estus, B. A. Millar, J. L. Carter, C. A. Wood and D. L. Bellinger. Bidirectional Communication between the Brain and the Immune System: Implications for Physiological Sleep and Disorders with Disrupted Sleep. Neuroimmunomodulation, vol. 13 pp. 357-374, 2006.

[74] Bluetooth. Bluetooth Technology Overview. [Online] Available: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/ [accessed 16.01.2022]

[75] Connectivity Standards Alliance. Zigbee Complete IOT solution. [Online] Available: https://csa-iot.org/all-solutions/zigbee/ [accessed 16.12.2021]

[76] ZWave Alliance. About Z-Wave Technology - Z-Wave Alliance. [Online] Available: https://z-wavealliance.org/about_z-wave_technology/ [accessed 16.01.2022]

[77] LoRA Alliance. What is LoRaWAN Specification - LoRA Alliance. [Online] Available: https://lora-alliance.org/about-lorawan/ [accessed 16.01.2022]

[78] OpenBSD. OpenBSD. [Online] Available: https://www.openbsd.org/ [accessed 16.01.2022]

[79] OpenSUSE. openSUSE - Linux OS. The makers' choice for sysadmins, developers and desktop users. [Online] Available: https://www.opensuse.org/ [accessed 16.01.2022]

[80] Microsoft. Explore Windows 11 OS, Computers, Apps, & More. [Online] Available: https://www.microsoft.com/en-us/windows [accessed 16.01.2022]

[81] Contiki NG. GitHub - contiki-ng/contiki-ng: Contiki-NG: The OS for Next Generation IoT Devices. [Online] Available: https://github.com/contiki-ng/contiki-ng [accessed 11.12.2021]

[82] TinyOS. GitHub - tinyos/tinyos-main: Main development repository for TinyOS (an OS for embedded, wireless devices). [Online] Available: https://github.com/tinyos/tinyos-main [accessed 11.12.2021]

[83] RIOT. RIOT - The friendly Operating System for the Internet of Things. [Online] Available: https://doc.riot-os.org/ [accessed 11.12.2021]

[84] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur and R. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, IETF, March 2012.

[85] IEEE Standards Association. IEEE 802.15.4-2020 - IEEE Standard for Low-Rate Wireless Networks. IEEE Computer Society, 2020.

[86] Bluetooth. Bluetooth Core Specification version 5.3. [Online] Available: https://www.bluetooth.com/specifications/specs/core-specification/ [accessed 16.12.2021]

[87] IEEE Standards Association. Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE Computer Society, 2011.

[88] Z. Shelby and C. Bormann. 6LoWPAN: The Wireless Embedded Internet, John Wiley & Sons Ltd, 2009.

[89] J. P. Vasseur and A. Dunkels. Interconnecting Smart Objects with IP: The Next Internet, Morgan Kaufmann Publishers, 2010.

[90] G. Montenegro, N. Kushalnagar, J. Hui and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, IETF, September 2007.

[91] A. Brandt, J. Buron and G. Porcu. Home Automation Routing Requirements in Low-Power and Lossy Networks. RFC 5826, IETF, April 2010.

[92] K. Pister, P. Thubert, S. Dwars and T. Phinney. Industrial Routing Requirements in Low-Power and Lossy Networks. RFC 5673, IETF, October 2009.

[93] J. Martocci, P. De Mil, N. Riou and W. Vermeylen. Building Automation Routing Requirements in Low-Power and Lossy Networks. RFC 5867, IETF, June 2010.

[94] M. Dohler, T. Watteyne, T. Winter and D. Barthel. Routing Requirements for Urban Low-Power and Lossy Networks. RFC 5548, IETF, May 2009.

[95] J. Hui and P. Thubert. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282, IETF, September 2011.

[96] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann. Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs). RFC 6775, IETF, November 2012.

[97] P. Thubert. IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Selective Fragment Recovery. RFC 8931, IETF, November 2020.

[98] P. Thubert. Routing for RPL (Routing Protocol for Low-Power and Lossy Networks) Leaves. RFC 9010, IETF, April 2021.

[99] J. Hui, D. Culler and S. Chakrabarti. 6LoWPAN: Incorporating IEEE 802.15.4 into the IP architecture, IPSO Alliance White Paper, 2009. [Online] Available: https://www.omaspecworks.org/wp-content/uploads/2018/03/6lowpan.pdf [accessed 16.12.2021]

[100] J. Olsson. 6LoWPAN Demystified, Texas Instruments, 2014. [Online] Available: https://www.ti.com/cn/lit/wp/swry013/swry013.pdf [accessed 16.12.2021]

[101] International Society of Automation. ISA100 Wireless Systems for Automation - ISA. [Online] Available: https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa100 [accessed 11.12.2021]

[102] J. Postel. User Datagram Protocol. RFC 768, IETF, August 1980.

[103] S. Deering and R. Hinden. Internet Protocol version 6 (IPv6) Specification. RFC 8200, IETF, July 2017.

[104] A. Conta, S. Deering and M. Gupta. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 4443, IETF, March 2006.

[105] J. Postel. Transmission Control Protocol. RFC 793, IETF, September 1981.

[106] C. Perkins, D. Johnson and J. Arkko. Mobility Support in IPv6. RFC 6275, IETF, July 2011.

[107] IETF. ROLL Workgroup Official Page. [Online] Available: https://datatracker.ietf.org/wg/roll/about/ [accessed 16.12.2021]

[108] J. P. Vasseur, M. Kim, K. Pister, N. Dejean and D. Barthel. Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks. RFC 6551, IETF, March 2012.

[109] P. Thubert. Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL). RFC 6552, IETF, March 2012.

[110] Internet Assigned Numbers Authority (iana). Internet Control Message Protocol version 6 (ICMPv6) Parameters. [Online] Available: https://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml [accessed on 05.11.2021]

[111] M. U. Farooq, M. Waseem, A. Khairi, and S. Mazhar. A Critical Analysis on the Security Concerns of Internet of Things (IoT). International Journal of Computer Applications, vol. 111, no. 7, February 2015.

[112] Z. -K. Zhang, M. C. Y. Cho, C. -W. Wang, C. -W. Hsu, C. -K. Chen and S. Shieh. IoT Security: Ongoing Challenges and Research Opportunities. IEEE 7th International Conference on Service-Oriented Computing and Applications, pp. 230-234, 2014.

[113] C. Ivan, M. Vujic, and S. Husnjak. Classification of security risks in the IoT environment. In Proceedings of the 26th International DAAAM Symposium on Intelligent Manufacturing and Automation, pp. 731-740, 2016.

[114] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. In Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, pp. 113-127, 2003.

[115] T. Tsao, R. Alexander, M. Dohler, V. Daza, A. Lozano and M. Richardson. A Security Threat Analysis for the Routing Protocol for Low-Power and Lossy Networks (RPLs). RFC 7416, IETF, January 2015.

[116] L. Wallgren, S. Raza and T. Voigt. Routing Attacks and Countermeasures in the RPL-Based Internet of Things. International Journal of Distributed Sensor Networks, 2013.

[117] National Institute of Standards and Technology (NIST). Advanced Encryption Standard (AES), Federal Information Processing Standards Publication (FIPS PUBS) 197, November 2001.

[118] National Institute of Standards and Technology (NIST). Elliptic Curve Cryptography - CSRC. [Online] Available: https://csrc.nist.gov/Projects/Elliptic-Curve-Cryptography [accessed 11.12.2021]

[119] R. L. Rivest. The RC5 encryption algorithm. In Preneel B. (eds) Fast Software Encryption (FSE 1994), Lecture Notes in Computer Science (LNCS), vol. 1008. Springer, 1995.

[120] Cryptopp. Cyrpto++ Library 8.6 - Free C++ class library of Cryptographic Schemes. [Online] Available: https://www.cryptopp.com/ [accessed 11.12.2021]

[121] Oracle Java. Java Cryptography Architecture (JCA) Reference Guide. [Online] Available: https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html [accessed 11.12.2021]

[122] PyCrypto. PyCrypto - The Python Cryptography Toolkit. [Online] Available: https://www.dlitz.net/software/pycrypto/ [accessed 11.12.2021]

[123] K. Ilgun. USTAT: a real-time intrusion detection system for UNIX. In Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, pp. 16-28, 1993

[124] K. Ilgun, R. A. Kemmerer and P. A. Porras. State transition analysis: a rule-based intrusion detection approach. IEEE Transactions on Software Engineering, vol. 21, no. 3, pp. 181-199, March 1995.

[125] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. In Proceedings of the 7th USENIX Security Symposium, San Antonio, Texas, January 26-29, 1998.

[126] AIDE. AIDE - Advanced Intrusion Detection Environment. [Online] Available: https://aide.github.io/ [accessed 11.12.2021]

[127] Contiki-NG. Documentation: Communication Security - contiki-ng/contiki-ng Wiki - GitHub. [Online] Available: https://github.com/contiki-ng/contiki-ng/wiki/Documentation:-Communication-Security [accessed 11.12.2021]

[128] Eclipse TinyDTLS. GitHub - eclipse/tinydtls: Eclipse tinydtls. [Online] Available: https://github.com/eclipse/tinydtls [accessed 11.12.2021]

[129] L. Casado and P. Tsigas. ContikiSec: A Secure Network Layer for Wireless Sensor Networks under the Contiki Operating System. NordSec 2009, Lecture Notes in Computer Science (LNCS), vol. 5838, pp.133-147, 2009.

[130] K. Krentz, H. Rafiee and C. Meinel. 6LoWPAN Security: Adding Compromise Resilience to the 802.15.4 Security Sublayer. In Proceedings of the International Workshop on Adaptive Security, 2013.

[131] R. Blom. An optimal class of symmetric key generation systems. Advances in Cryptology, EUROCRYPT 1984, Lecture Notes in Computer Science (LNCS), vol. 209, pp. 335-338, 1985.

[132] C. Y. Chen and H. C. Chao. A survey of key distribution in wireless sensor networks. Security Communication Networks, vol. 7, no. 12, pages 2495-2508, 2014.

[133] H. Chan, A. Perrig and D. Song. Random Key Predistribution Schemes for Sensor Networks. In Proceedings of the 2003 IEEE Symposium on Security and Privacy, Page 197-217, 2003.

[134] S. Zhu, S. Setia and S. Jajodia. LEAP+: efficient security mechanisms for large-scale distributed sensor networks. ACM Transactions on Sensor Networks, vol. 2, no. 4, pp. 500-528, 2006.

[135] A. Perrig, R. Szewczyk, V. Wen, D. Culler and J.D. Tygar. SPINS: Security Protocols for Sensor Networks. Mobile Computing and Networking, 2001.

[136] F. Busching, A. Figur, D. Schurmann and L. Wolf. Utilizing Hardware AES Encryption for WSNs. In Proceedings of 10th European Conference on WSNs, Belgium, 2013.
[137] T. Grandison and E. Terzi. Intrusion Detection Technology. In Liu L., Özsu M. (eds) Encyclopedia of Database Systems, 2017, Springer.
[138] J. S. Sherif and T. G. Dearmond. Intrusion Detection: Systems and Models. In Proceedings of the 11th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE), 2002.
[139] F. Liu, X. Cheng and D. Chen. Insider Attacker Detection in Wireless Sensor Networks. In Proceeding of the 26th IEEE International Conference on Computer Communications (IEEE INFOCOM), pp. 1937-1945, 2007.
[140] S. Raza, L. Wallgren and T. Voigt. SVELTE: Real-time intrusion detection in the Internet of Things. Ad Hoc Networks, vol. 11, no. 8, pp. 2661-2674, 2013.
[141] I. Krontiris, Z. Benenson, T. Giannetsos, F. C. Freiling, and T.Dimitriou. Cooperative Intrusion Detection in Wireless Sensor Networks. In Roedig U., Sreenan C.J. (eds) Wireless Sensor Networks (EWSN), Lecture Notes in Computer Science (LNCS), vol. 5432, pp. 263-278, 2009.
[142] P. Pongle and G. Chavan. Real time intrusion and wormhole attack detection in internet of things. International Journal of Computer Applications, vol. 121, no. 9, pp. 1-9, 2015.
[143] A. Le, J. Loo, K. K. Chai, and M. Aiash. A specification-based IDS for detecting attacks on RPL-based network topology. Information, vol. 7, no. 2, 2016.
[144] J. P. Amaral, L. M. Oliveira, J. J. P. C. Rodrigues, G. Han and L. Shu. Policy and network-based intrusion detection system for IPv6-enabled wireless sensor networks. In Proceedings of the IEEE International Conference on Communications (ICC), pp. 1796-1801, 2014.
[145] Apple. iOS 15 - Apple. [Online] Available: https://www.apple.com/ios/ios-15/ [accessed 21.12.2021]
[146] Sailfish OS. Sailfish OS fourth generation. [Online] Available: https://sailfishos.org/ [accessed 21.12.2021]
[147] KaiOS. Home - KaiOS. [Online] Available: https://www.kaiostech.com/ [accessed 21.12.2021]
[148] Amazon. Fire OS Overview - Amazon Fire TV. [Online] Available: https://developer.amazon.com/docs/fire-tv/fire-os-overview.html [accessed 21.12.2021]
[149] Lineage. LineageOS Android Distribution. [Online] Available: https://lineageos.org/ [accessed 21.12.2021]
[150] Android. Android Open Source Project. [Online] Available: https://source.android.com/ [accessed 16.12.2021]
[151] Google. Google Play. [Online] Available: https://play.google.com/store [accessed 16.12.2021]
[152] Google. Android - Google Mobile Services. [Online] Available: https://www.android.com/gms/ [accessed 16.12.2021]
[153] Android. Kernel - Android Open Source Project. [Online] Available: https://source.android.com/devices/architecture/kernel [accessed 16.12.2021]
[154] Android. HAL Types - Android Open Source Project. [Online] Available: https://source.android.com/devices/architecture/hal-types [accessed 16.12.2021]
[155] Android. Download Android Studio and SDK tools - Android Developers. [Online] Available: https://developer.android.com/studio/ [accessed 16.12.2021]
[156] AppBrain. Number of Android applications on the Google Play store. [Online] Available: https://www.appbrain.com/stats/number-of-android-apps [accessed 16.12.2021]
[157] Statista. Google Play Store: number of apps 2021. [Online] Available: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/ [accessed 16.12.2021]
[158] PalmSource Inc. OpenBinder. [Online] Available: http://www.angryredplanet.com/~hackbod/openbinder/docs/html/ [accessed 16.12.2021]
[159] Android. Manifest.permission - Android Developers. [Online] Available: https://developer.android.com/reference/android/Manifest.permission.html [accessed 16.12.2021]

[160] Android. Permissions on Android - Android Developers. [Online] Available: https://developer.android.com/guide/topics/permissions/overview [accessed 16.12.2021]

[161] Android. Security Enhancements - Android Open Source Project. [Online] Available: https://source.android.com/security/enhancements [accessed 16.12.2021]

[162] M. Hypponen. Malware goes mobile. Scientific American, vol. 295 no.5, pp. 70–77, 2006.

[163] S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer and A. -R. Sadeghi. Xmandroid: A New Android Evolution to Mitigate Privilege Escalation Attacks. Technische Universität Darmstadt, Technical Report TR-2011-04.

[164] M. Conti, B. Crispo, E. Fernandes and Y. Zhauniarovich. CRêPE: A System for Enforcing Fine-Grained Context-Related Policies on Android. IEEE Transactions on Information Forensics and Security, vol. 7, no. 5, pp. 1426-1438, 2012.

[165] S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A. -R. Sadeghi and B. Shastry. Practical and lightweight domain isolation on android. In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, pp. 51-62, 2011.

[166] G. Russello, M. Conti, B. Crispo, and E. Fernandes. MOSES: supporting operation modes on smartphones. In Proceedings of the 17th ACM symposium on Access Control Models and Technologies (SACMAT), pp. 3-12, 2012.

[167] G. Russello, M. Conti, B. Crispo, E. Fernandes and Y. Zhauniarovich. Demonstrating the effectiveness of moses for separation of execution modes. In Proceedings of the 2012 ACM conference on Computer and communications security, pp. 998-1000, 2012.

[168] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B. -G. Chun, L. P. Cox, J. Jung, P. McDaniel and A. N. Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. ACM Transactions on Computer Systems (TOCS) vol. 32, no.2, 2014.

[169] S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, A. -R. Sadeghi and B. Shastry. Towards Taming Privilege-Escalation Attacks on Android. In Proceedings of the 19th Annual Network and Distributed System Security Symposium (NDSS), The Internet Society, 2012.

[170] S. Bugiel, S Heuser and A. -R. Sadeghi. Flexible and fine-grained mandatory access control on android for diverse security and privacy policies. In Proceedings of the 22nd USENIX Security Symposium (USENIX Security), pp.131-146, 2013.

[171] M. Miettinen, S. Heuser, W. Kronz, A. -R. Sadeghi, and N. Asokan. Conxsense: automated context classification for context-aware access control. In Proceedings of the 9th ACM symposium on Information, computer and communications security, pp. 293-304, 2014.

[172] Y. Wang, K. Vangury, and J. Nikolai. Mobileguardian: A security policy enforcement framework for mobile devices. In Proceedings of IEEE International Conference on Collaboration Technologies and Systems (CTS), pp. 197-202, 2014.

[173] S. Heuser, A. Nadkarni, W. Enck and A. -R. Sadeghi. Asm: A programmable interface for extending android security. In Proceedings of the 23rd USENIX Security Symposium (SEC), 2014.

[174] M. Backes, S. Bugiel, S. Gerling, and P. von Styp-Rekowsky. Android security framework: extensible multi-layered access control on Android. In Proceedings of the 30th ACM Annual Computer Security Applications Conference (ACSAC), pp. 46-55, 2014.

[175] X. Wang, K. Sun, Y. Wang and J. Jing. Deepdroid: Dynamically enforcing enterprise policy on android devices. In Proceedings of the 22nd Annual Network and Distributed System Security Symposium (NDSS), The Internet Society, 2015.

[176] R. Xu, H. Saidi and R. Anderson. Aurasium: Practical policy enforcement for android applications. In Proceedings of the 21st USENIX Security Symposium (USENIX Security), pp. 539-552, 2012.

[177] J. Jeon, K. K. Micinski, J. A. Vaughan, A. Fogel, N. Reddy, J. S. Foster and T. Millstein. Dr. android and Mr. hide: fine-grained permissions in android applications. In Proceedings of the 2nd ACM workshop on Security and privacy in smartphones and mobile devices, pp. 3-14, 2012.

[178] G. Russello, A. B. Jimenez, H. Naderi and W. van der Mark. Firedroid: hardening security in almost-stock android. In Proceedings of the 29th ACM Annual Computer Security Applications Conference, pp. 319-328, 2013.

[179] Y. Wang, Y. Chen, F. Ye, J. Yang and H. Liu. Towards understanding the advertiser's perspective of smartphone user privacy. In Proceedings of the 35th IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 288-297, 2015.

[180] I. Ganchev, Z. Ji and M. O'Droma. A generic IoT architecture for smart cities. In Proceedings of 25th IET Irish Signals & Systems Conference and China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014), pp. 196-199, 2014.

[181] Z. Yang, Y. Yue, Y. Yang, Y. Peng, X. Wang and W. Liu. Study and application on the architecture and key technologies for IOT. 2011 International Conference on Multimedia Technology, pp. 747-751, 2011.

[182] L. R. Salazar, A. R. G. Ramirez and V. R. Q. Leithard. Development of an IoT Architecture for an Electronic Cane. In Proceedings of the 14th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1-4, 2019.

[183] A. Ordonez-García, M. Siller and O. Begovich. IoT architecture for urban agronomy and precision applications. In Proceedings of the IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), pp. 1-4, 2017.

[184] C. -L. Zhong, Z. Zhu and R. -G. Huang. Study on the IOT Architecture and Gateway Technology. In Proceedings of the 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), pp. 196-199, 2015.

[185] J. Zhen. A Novel Two-Layer Decentralized Ledger Architecture for Internet of Things. In Proceedings of IEEE Global Communications Conference (GLOBECOM), pp. 1-6, 2019.

[186] R. S. Pressman and B.R. Maxim. Software Engineering - A Practitioner's Approach, Eighth Edition. McGraw-Hill Education, 2015.

[187] IEEE Standards. ISO/IEC/IEEE Systems and software engineering – Architecture description. ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000), pp. 1-46, December 2011.

[188] R. Ross, M. McEVilley and J. C. Oren. Systems Security Engineering - Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems. National Institute of Standards and Technology, Special Publication 800-160, vol. 1, November 2016.

[189] IEEE Standards Association. IEEE Standard for an Architectural Framework for the Internet of Things (IoT). IEEE Standard 2413-2019, pp. 1-269, 10 March 2020.

[190] R. Shirey. Internet Security Glossary version 2. RFC 4949, IETF, August 2007.

[191] Joint Task Force. Security and Privacy Controls for Information Systems and Organizations. National Institute of Standards and Technology, Special Publication 800-53, Revision 5, September 2020.

[192] S. Cobianchi, L. Casals-Diaz, J. Jaramillo and X. Navarro. Differential effects of activity dependent treatments on axonal regeneration and neuropathic pain after peripheral nerve injury. Experimental Neurology, vol. 240, pp. 157-167, 2013.

[193] A. W. English, J. C. Wilhelm and P. J. Ward. Exercise, neurotrophins, and axon regeneration in the PNS. Physiology, vol. 29, no. 6, pp. 437-445, 2014.

[194] Psichologyanswers.com. Can damaged neurons be reparied? [Online] Available: https://psichologyanswers.com/library/lecture/read/253277-can-damaged-neurons-be-repaired [accessed 16.12.2021]

[195] T. Gordon and A. W. English. Strategies to promote peripheral nerve regeneration: electrical stimulation and/or exercise. The European journal of neuroscience, vol. 43, no. 3, pp. 336-350, 2016.

[196] R. J. M. Franklin, and C. ffrench-Constant. Remyelination in the CNS: from biology to therapy. Nature Reviews Neuroscience, vol. 9, no. 11, pp. 839-855, 2008.

[197] A. Tedeschi, T. Omura and M. Costigan. CNS repair and axon regeneration: Using genetic variation to determine mechanisms. Experimental Neurology, vol. 287, part 3, pp. 409-422, 2017.

[198] M. Berry, Z. Ahmed and A. Logan. Return of function after CNS axon regeneration: Lessons from injury-responsive intrinsically photosensitive and alpha retinal ganglion cells. Progress in Retinal and Eye Research, vol. 71, pp. 57-67, 2019.

[199] B. P. Liu, W. B. Cafferty, S. O. Budel, and S. M. Strittmatter. Extracellular regulators of axonal growth in the adult central nervous system. Philosophical transactions of the Royal Society of London, Series B, Biological sciences, vol. 361, no. 1473, pp. 1593-1610, 2006.

[200] S. Okada, M. Hara, K. Kobayakawa, Y. Matsumoto and Y. Nakashima. Astrocyte reactivity and astrogliosis after spinal cord injury. Neuroscience Research, vol. 126, pp. 39-43, 2018.

[201] A. Begega, P. Alvarez-Suarez, P. Sampedro-Piquero and M. Cuesta. Chapter 1 - Effects of Physical Activity on the Cerebral Networks. Physical Activity and the Aging Brain, Academic Press, pp. 3-11, 2017.

[202] G. Kempermann. Neurogenesis in the Intact Adult Brain. Encyclopedia of Neuroscience, Academic Press, pp. 443-447, 2009.

[203] National Institute of Neurological Disorders and Stroke. Brain Basics: The Life and Death of a Neuron. [Online] Available: https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Life-and-Death-Neuron [accessed 19.12.2020]

[204] Y. Hatanaka, Y. Zhu, M. Torigoe, Y. Kita, F. Murakami. From migration to settlement: the pathways, migration modes and dynamics of neurons in the developing brain. Proceedings of the Japan Academy, Series B, vol. 92, no. 1, pp. 1-19, 2016.

[205] M. Kim, B. Bjorke and G. S. Mastick. Motor neuron migration and positioning mechanisms: New roles for guidance cues. Seminars in Cell & Developmental Biology, vol. 85, pp. 78-83, 2019.

[206] Y. Rao, K. Wong, M. Ward, C. Jurgensen, and J. Y. Wu. Neuronal migration and molecular conservation with leukocyte chemotaxis. Genes & development, vol. 16, no. 23, pp. 2973-2984, 2002.

[207] Z. Chen. Common cues wire the spinal cord: Axon guidance molecules in spinal neuron migration. Seminars in Cell & Developmental Biology, vol. 85, pp. 71-77, 2019.

[208] S. Hiller-Sturmhöfel and A. Bartke. The endocrine system: an overview. Alcohol health and research world, vol. 22, no. 3, pp. 153-164, 1998.

[209] B. M. Carlson. Chapter 9 - The Endocrine System. The Human Body, Academic Press, pp. 241-269, 2019.

[210] SEER Training Modules - National Cancer Institute. Introduction to the Cardiovascular System. [Online] Available: https://training.seer.cancer.gov/anatomy/cardiovascular/ [accessed 19.12.2020]

[211] E. A. MacDonald, R. A. Rose and T. A. Quinn. Neurohumoral Control of Sinoatrial Node Activity and Heart Rate: Insight From Experimental Models and Findings From Humans. Frontiers in Physiology, vol. 11, 2020.

[212] R. Gordan, J. K. Gwathmey and L.H. Xie. Autonomic and endocrine control of cardiovascular function. World Journal of Cardiology, vol. 7, no. 4, pp. 204-214, 2015.

[213] S. Narain, T. D. Vo-Huu, K. Block and G. Noubir. Inferring User Routes and Locations Using Zero-Permission Mobile Sensors. IEEE Symposium on Security and Privacy (SP), pp. 397-413, 2016.

[214] J. R. Kwapisz, G. M. Weiss and S. A. Moore. Activity Recognition Using Cell Phone Accelerometers. SIGKDD Explorations Newsletter, vol. 12, no. 2, pp. 74-82, 2011.

[215] S. Nawaz and C. Mascolo. Mining Users' Significant Driving Routes with Low-Power Sensors. In Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems, pp. 236–250, 2014.

[216] A. Wang, G. Chen, J. Yang, S. Zhao and C. -Y. Chang. A Comparative Study on Human Activity Recognition Using Inertial Sensors in a Smartphone. IEEE Sensors Journal, vol. 16, no. 11, pp. 4566-4578, 2016.

[217] Android. Permission Checker - Android developers. [Online] Available: https://developer.android.com/reference/androidx/core/content/PermissionChecker [accessed 18.12.2021]

[218] N. K. Thanigaivelan, E. Nigussie, S. Virtanen and J. Isoaho. Towards Human Bio-Inspired Defence Mechanism for Cyber Security. IEEE Security and Privacy Workshops (SPW), pp. 276-280, 2018.

[219] N. K. Thanigaivelan, E. Nigussie, R. K. Kanth, S. Virtanen and J. Isoaho. Distributed internal anomaly detection system for Internet-of-Things. 13th IEEE Annual Consumer Communications & Networking Conference (CCNC), pp. 319-320, 2016.

[220] N. K. Thanigaivelan, E. Nigussie, S. Virtanen and J. Isoaho. Hybrid Internal Anomaly Detection System for IoT: Reactive Nodes with Cross-Layer Operation. Security and Communication Networks, vol. 2018, Article ID 3672698, 15 pages, 2018.

[221] N. K. Thanigaivelan, E. Nigussie, S. Virtanen and J. Isoaho. Conceptual Security System Design for Mobile Platforms Based on Human Nervous System. In Barolli L., Xhafa F., Hussain O. (eds) Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Advances in Intelligent Systems and Computing (AISC), vol. 994, 2019, Springer.

[222] N. K. Thanigaivelan, E. Nigussie, A. Hakkala, S. Virtanen and J. Isoaho. CoDRA: Context-based dynamically reconfigurable access control system for android. Journal of Network and Computer Applications (JNCA), vol. 101, pp. 1-17, 2018.

[223] R. L. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, vol. 21, no. 2, pp. 120-126, 1978.

TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

Painosalama, Turku, Finland 2023

ISBN 978-951-29-9196-9 (PRINT)
ISBN 978-951-29-9197-6 (PDF)
ISSN 2736-9390 (Painettu/Print)
ISSN 2736-9684 (Sähköinen/Online)