International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:4, 2015

A Robust Implementation of a Building Resources Access Rights Management System

E. Neagoe, V. Balanica

***Abstract*—**A Smart Building Controller (SBC) is a server software that offers secured access to a pool of building specific resources, executes monitoring tasks and performs automatic administration of a building, thus optimizing the exploitation cost and maximizing comfort. This paper brings to discussion the issues that arise with the secure exploitation of the SBC administered resources and proposes a technical solution to implement a robust secure access system based on roles, individual rights and privileges (special rights).

***Keywords*—**Access authorization, smart building controller, software security, access rights.

I. Introduction

IN the Smart Building Control research and development field, and further in the present document, an *environment* is considered to be the entire composition of the (BACnet standard complying [1], and alike) building devices that are subscribed and registered to a central monitoring and administrative center called a SBC. The SBC (Smart Building Controller) is a server software that offers secured access to a pool of building operation specific resources [2], defined in this paper as *SBC objects*, i.e. hardware, software or abstract [3]–[5]. A hardware object is a physical device, available in the local SBC environment and being in direct communication with it, described by a set of reportable and/or configurable properties. HVAC specific components, sensors, security cameras and other monitoring devices, automatic access (doors) and watering systems, RFID readers, multimedia appliances, internet access points and communication infrastructure are examples of local available hardware. These devices, if digitally controllable using a communication protocol over a data transmission network are directly manageable by an artificial intelligent management system as a SBC. Other kind of resources could be found outside the physical SBC managed environment but added to the local resources through a published interface, for instance an automated earthquake alarm, a weather casting service or a mail server. The third category of SBC objects are abstract and directly responsible of the SBC AI (Artificial Intelligence) managed actions. Here we place conditions, actions, schedules, triggers, policies, patterns, procedures, all logical bits and bytes that put together build-up the SBC intelligent behavior that pursues the optimization of the building's exploitation cost while enhancing the user's safety and comfort. All SBC objects will be further indicated also as "nodes" because the convenient representation of the SBC environment as a graph.

The access, administration and supervision of the SBC resources are performed using a standalone client or through a browser presented GUI (Graphical User Interface). In this context, a control session is the legitimate usage of the available SBC resources in the timeframe starting with the user's secured login and ending at logout or session timeout, only within the user's administrative role. The authenticated and lawful access to environments resources is of paramount importance since the goal is to refine, optimize and enhance the building's exploitation by adding intelligent features, not to establish dangerous property and privacy attack vectors, [6].

This paper analyses the requirements of an integrated authority management system in the field of smart building control architecture [7], and secondly it describes the implementation of a secure robust access model with minimal impact on the SBC environment operational performance [8], [9].

II. Generic Requirements of Resources Secure Access Model

A major challenge in SBC administration is managing the complex security requirements. The unauthorized access to the reporting capabilities of environment resources (let alone their control), can provide an attacker with sensitive information, allowing a more sophisticated attack, premises breach or simply compromising privacy. Most of the network communication will carry M2M (machine-to-machine) requests and messages therefore each SBC unit must detain a trustee database, along with each partner rights to request information and to order actions.

Any security model must be robust in order to withstand tampering. With the increased complexity of interacting actors, the security layer must try to cope with the unpredictable environment change as much as possible [10]. In the scope of this paper we define robustness to be the model's property to avoid a single point-of-failure design by storing the legitimate utilization rights as close to the execution point as possible (redundant distributed database), to exhibit a gracefully operation degradation in case of node, actors, catastrophic failure, to seamlessly accommodate the addition or removal of nodes. Because each node detains in its trustee list only information regarding its accredited partners, a compromised node does not affect the whole environment. This characteristics support the proposed security model "robust" claim.

Eugen Neagoe and Victor Balanica are with the Dept. of Automatic Control and System Engineering, University Politehnica of Bucharest, Romania, e-mail: (vidda.loca@gmail.com, vicord20011@gmail.com).

Systems with increased complexity must be designed with mechanisms to handle failure and security as a core issue. We aim to avoid possible failures coming from complexity and obscurity through simplicity, without loss in capabilities. This approach is consistent from the implementation to maintenance and utilization.

The proposed model offers both Role-Based Access Control (RBAC) [11]-[14], and Discretionary Access Control (DAC) [15], effectively addressing the role hierarchy issue, rights inheritance and privileged access rights outside the administrative chain. The accent is put on a role-based paradigm due to facilities offered in complex system administration, [16]. In order to permit a clear and rapid understanding of a user's access rights extent, to avoid misrepresentation and potential obscure implications in complex environments and large administrative personal, our model forbids a role to contain one or more other roles.

Our solution to the relationship among administrative roles is by hierarchy and inheritance. Mainly, permissions are established for roles to which users are assigned, following the concept of restricted hierarchical RBAC [12], further constraints being possible through the node's internal logic (operational policies). Mandatory Access Control (MAC) [16], capabilities are implemented through "Special Rights". This feature adds non-inheritable rights that override de hierarchical inherited ones [17], effectively enabling security configuration at node feature level and mitigating fault hierarchies [18].

In the scope of this document, a node feature is considered any reportable or configurable node state. For instance, an air-conditioning device can report room temperature (sensor state - read only) and modify it (actuator state - set desired temperature).

III. SBC Management Design Requirements

While developing a security framework for the SBC operations the following requirements have been taken into consideration, [19]-[21]:

- *Database*. All information related to users and SBC resources access rights must be available through a configuration interface and locally stored in a compact data structure. The state-of-the-art solution is an encrypted database.
- *Encryption*. Given the sensitive nature of exchanged information, connection-based designs and encrypted communications channels are preferable even inside the SBC network. For SBC access performed over internet, tunneling techniques should be implemented.
- *Check Point*. A SBC management session (login) should be permitted only after the SBC validates the user's credentials against its database.
- *Single Session*. A single SBC management session should be permitted per user at any time, with limited lifespan.
- *Limited View*. Any user should view and manage only the SBC objects that were made available to read, write or delete through some sort of permissions granting mechanism.
- *Roles*. Role-based access rights configuration and administrative group structures must be available to the SBC manager, along with individual rights management.
- *Special Permissions*. A mechanism to circumvent or override the automatic hierarchical inheritance of rights should be made available in order to permit rights and tasks delegation with utmost granularity.
- *Delegation*. The administrative possibility to spawn or administer role or user rights in the range of the detained attributions, with multi-level grant and grant independent revocation, [15].

The description of the present integrated secure management model is based on various practical considerations [22], [23]:

- The database operational overhead should be minimal and must provide secured access for any set of managed SBC objects or services. The implementation should be as simple and robust as possible, in order to avoid unforeseen consequences.
- The available resources should be administered by managerial hierarchal roles, a manager being capable to delegate his rights, suspend or deny access to any lower rank than his.
- No user or administrator should be able to configure his own rights by any means.
- Every SBC object should require specific minimal access rights, being under the administration of every higher managerial authority and out of reach for the lesser ones.
- There should be a possibility to grant limited and isolated rights in a certain environment, for instance given a device, a "manual only" schedule setup could be in the exclusive duty of a certain user, even if the user has no other rights in the environment. This functionality would render the system more flexible by adding the possibility of user identity well bounded scenarios.
- Each user should be able to access only the resources that are made available to him by properly setting the SBC objects requirements, the user's generic access rights or by special rights issued for designated objects.
- Every SBC object has to have a "disabled" state for maintenance and testing purposes, as well as a "manual only" state that will disable automatic object management in pursue of an abstract/complex goal. In such a state the object will be managed by user defined actions alone.
- The security system should be extendable even to device properties as in "on/off" for a light bulb. It should be possible for an environment to be configured in such a manner that ONLY a certain user (not even a higher authority user) may switch a device property on/off.
- The default supreme managerial authority cannot be deleted or limited in any way.

As the internet is used in SBC operational management and control, specific precautions measures should be taken into consideration. The main identifiable attack vectors are: Denial of Service, [24], Man in the Middle, [25]–[28], Spoofing, [29], Eavesdropping. Due to SBC specificity, the information obtained by an attacker could be used in preparation of further

attacks (meshing), [6], [30]. For instance, the reported history of a presence sensor system could provide valuable information about the timeframes when the building is more vulnerable to theft.

IV. The Three-Authority Management (TAM) Model

Pursuing the above stated requirements, we propose the following dedicated SBC secure access model, named the Three-Authority Management (TAM). The name indicates the three access types this model uses to manage a SBC object:

READ – the right or capability a user has to be aware of the existence of a certain SBC object and to view reports about its current state.

WRITE – the user's right or capability to change a certain SBC object configurable properties or parameters. In order to have this possibility, the READ right over that object is mandatory.

DELETE – the right or capability to exclude a certain SBC object from the SBC administration. The deletion in effect determines the SBC command center to "forget" about a physical device by erasing its records or to effectively delete a local software object (script) like a schedule or policy for instance. READ and WRITE rights are mandatory for this kind of action.

These three capabilities express the authority range that a specified user has over a particular SBC resource. In order to avoid the need of n*m distinct entries in a security table that describes relationships between n users and m SBC objects, we express the user's rights and the object's requirements as a three byte values array:

$$\text{USER}:\{[\boldsymbol{byte}(0-255)_{READ}],[\boldsymbol{byte}(0-255)_{WRITE}],[\boldsymbol{byte}(0-255)_{DELETE}]\}$$
$$\text{RESOURCE}:\{[\boldsymbol{byte}(0-255)_{READ}],[\boldsymbol{byte}(0-255)_{WRITE}],[\boldsymbol{byte}(0-255)_{DELETE}]\}$$

Thus, a user's administrative rights will be expressed as a three byte values string linked together with dashes: READ-WRITE-DELETE, ex. 10-9-8. The same formula is used to indicate the object's security requirements.

The granted rights are computed by comparing the bytes expressing the user's maximal access rights with the object's minimal access requirements, and expressed as Boolean values: CanView, CanEdit, CanDelete:

$$CanView_{bool} = (UserMaxRead_{byte} \geq ResourceMinRead_{byte})?:true|false$$
$$CanEdit_{bool} = (UserMaxWrite_{byte} \geq ResourceMinWrite_{byte})?:true|false$$
$$CanDelete_{bool} = (UserMaxDelete_{byte} \geq ResourceMinDelete_{byte})?:true|false$$

In our approach, a user has generic access rights and an object (SBC resource), specific minimal rights requirements. Only if the user's generic rights match or exceed the objects requirements the access type is granted.

To enforce logic in the way SBC resources are managed and to permit complex administrative workflow scenarios, the following must be enforced:

$$User_{MaxRead} \geq User_{MaxWrite} \geq User_{MaxDelete}$$
$$Object_{MaxRead} \leq Object_{MaxWrite} \leq Object_{MaxDelete}$$

As a consequence, DELETE rights include all WRITE rights and WRITE rights include the READ right.

The reason is that a user must have "View" access before he can "Edit" an object and must have "Edit" rights in order to be capable to "Delete". The same is true for an object but in reverse as we express the object's requirements: The "Delete" requirements should be greater than "Edit" or "Read" ones. The logic of the secured access mechanism of *TAM* implements the principle the SBC Objects and the SBC Users should have a complementary functionality.

This implementation of user rights and objects requirements system has the advantages of an inheritable hierarchy of administrative roles but lacks in establishing granular rights. For instance, if an object requires at least 100-100-100 administrative rights, any user that is configured to manage that object will automatically inherit rights over any lesser rights object, for example 99-99-99. To solve this inconvenient we added a specific user-object administrative rights tracking system called *SpecialRights*, that overrides the *TAM* computed rights with an arbitrary configuration for a particular set{user_ID, object_ID}. This hybrid approach permits to configure administrative roles that have in effect no inherited rights but only a fixed, constant, granular set, regardless of what other resources are added later in the administrative process.

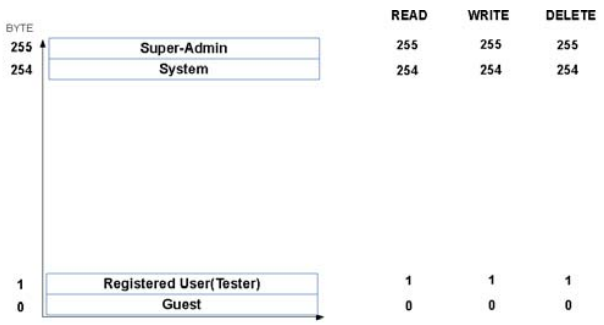

Fig. 1 Administrative authority roles with default Read, Write and Delete values

We assume that the environment complexity will not exceed the 256 hierarchical levels of administration / authority levels a byte can express. Each authority level is treated like a role that can be assigned to an unlimited number of members/users/managers. Thus, the TAM (Three Authority Management) security model permits enough authority levels for the main access types (read, write and delete) that permit even in very large organizations an appropriate hierarchical control configuration. If necessary, the model can be designed to permit more role levels by simply extending the integer range. The model is thought with a set of default roles: Super-Admin, System, Registered-User (Tester) and Guest with default access rights (see Fig. 1).

Because automatically triggered actions should be possible in a SBC environment, the special role SYSTEM has lesser administrative rights than the SUPER-ADMIN account, by default. This layout is necessary in order to permit the

existence of at least a class of SBC objects outside the automatically SYSTEM's management capabilities. In this paper we arbitrarily set the SYSTEM's maximal access rights at 253 in order to underline the importance of this role in the SBC AI management. The actual value is left at the developer's decision in the model implementation phase.

MinRead	0	Default SBC object:
MinWrite	1	• Configured with gradual access requirements
MinDelete	2	

MaxRead	0	Guest:
MaxWrite	0	• Can view the default SBC object
MaxDelete	0	• Cannot edit, nor delete the object

MaxRead	1	Registered user:
MaxWrite	1	• Can view and edit the default SBC object
MaxDelete	1	• Cannot delete

MaxRead	255	Super-Admin:
MaxWrite	255	• Can view, edit and delete any SBC object, in this example, the default one.
MaxDelete	255	

Fig. 2 Example of computed TAM rights for different default roles

The SBC TAM security descriptors will be stored separately, in 3 Security Descriptor tables named: *UserTAM*, *ObjectTAM* and *SpecialRights*.

The SBC access granting workflow:

- Receives the provided user credentials and checks them against its database
- If the provided credentials do check-out, the user's maximal access rights are read from the UserTAM table
- The user's rights are used to SELECT out of the ObjectTAM table every object that the user has at least READ rights for (CanView).
- The query result is formatted into an XML structure, including only the user's access rights that are evaluated as TRUE: CanView, CanEdit, CanDelete. The rights order is progressive, as enumerated.
- The login timestamp is recorded, an expiring session token is issued for that user and the previously formatted XML is returned to the user's administrative client (browser or standalone application).

In the following, as illustrated in Fig. 2, we will refer to the *TAM* patterns as a sequence of 3 numbers linked by dashes, representing for a user the *Max*Read-*Max*Write-*Max*Delete rights and the *Min*Read-*Min*Write-*Min*Delete access requirements for an object.

In this example a registered user has no rights to create objects, because a SBC object creation can inherit at most the maximal rights of its creator, which in this case do not match the 0-1-2 default minimal requirements. The default object requirements as well as the "Guest" or "Registered user" role access rights can be changed according to the implemented scenario.

A. The UserTAM Security Descriptor

Each SBC user has a security clearance, expressed (as seen also above in the SBC object section) as a sequence of the three authority levels (Read-Write-Delete).

Fig. 3 presents the database implementation of the UserTAM table. The *MaxRead*, *MaxWrite*, *MaxDelete* columns store the byte values of the user's maximal access rights. The access rights are expressed through integers between 0-255 and define the maximal object level a user can view and modify.

Index	Object User ID	Max Read	Max Write	Max Delete	Disabled	Locked
int	bigint	byte	byte	byte	bool	bool

Fig. 3 UserTAM table layout

The enforced rule is: $MaxRead \geq MaxWrite \geq MaxDelete$.

By default, a new added user is created with the "Registered user" access rights (1-1-1). The model policy is to promote users from the "Registered" state, not to demote them.

B. User Flags

Extending the UserTAM table functions (seen in Fig. 6), the user records present two additional columns: the "Disabled" and the "Locked" boolean flags. These descriptors have the following meaning:

- *Disabled*. When this flag is set (TRUE) the user is prevented from any SBC usage or administration. His credentials will not permit the login, regardless of the delegated TAM role at the moment of suspension. However, all settings and configuration related to him and the existing SBC environment are preserved, awaiting other administrative actions. The SUPERUSER account cannot be disabled, even by SUPERUSER himself.
- *Locked* (Read Only). While the LOCKED flag is set (TRUE), the user's TAM rights are preserved in the system's configuration but any TAM inherited WRITE or DELETE rights will be ignored by default. This flag has the function to permit the delegation of specific rights to a certain user through the SpecialRights mechanism and to avoid the standard TAM access levels and inheritance mechanism. The removal of the "Locked" flag re-enables the TAM hierarchical roles.

C. The ObjectTAM security descriptor

The ObjectTAM table implements the MinDelete, MinWrite, MinRead columns that express the minimal authority level required by an object for each of the three access types, as described in Fig. 4. The required access levels are expressed through a byte value (0-255).

In-dex	Object Unique ID	Max Read	Max Write	Max Delete	Dis-abled	Locked	Manual Only
int	bigint	byte	byte	byte	bool	bool	bool

Fig. 4 ObjectTAM table layout

The enforced rule is: $MinRead \leq MinWrite \leq MinDelete$.

D. Object Flags

Extending the ObjectTAM table functionality, a series of

boolean flags are added to the object's security descriptor as detailed in Fig. 5:

Disabled	Locked	Manual Only
bool	bool	bool

Fig. 5 Object flags layout

Flags description:

- *Disabled*. With this flag set, the existence of an object is reported only by displaying a grayed out name of the object in the GUI, without any parameters. In this state the SYSTEM will not be able to access the object. Any readings pushed from the object will be ignored. Policies that strictly depend on that object will be suspended.
- *Locked* (Read Only). Setting the LOCKED flag (true) disables the TAM inherited WRITE and DELETE rights, retaining the READ rights hierarchically enabled for the all *0-253* levels. The only roles that retain full access over the object remain 254 – System and 255 –Super-Admin. In effect the locked flag resembles an object with 0-254-254 rights.

The LOCKED flag is useful in order to achieve a high granular management configuration for some specific SBC objects. Setting the LOCKED flag true and configuring special security rights (the *Special Rights* table) for one or more users effectively delegates exclusive management rights over that object.

- *Manual Only* (Disable SYSTEM override). While pursuing the configured goals, the "System" account (SBC server A.I.) is supposed to exercise an intelligent control over the available resources and in the process to change (edit) the object's parameters. That is for example, to shut-down the TV set when nobody is watching. However, in order to forbid the "System" to automatically control some resources, this flag must be set (true), meaning that only the actions configured by the user will be executed and nothing more (the SBC A.I. generated behavior will ignore the object).

E. Special Rights

While the principal mechanism of TAM provides a hierarchical management mechanism, special use case scenarios demand a way to ignore the inheritance chain and to grant only limited, user-object nominal rights. This is the reason we introduced an additional method to achieve this behavior. The model is able to bypass and override the main rights inheritance mechanism in order to grant exceptional rights with the aid of so called "SpecialRights".

The special rights are nominal established rights for a specific user over a certain object. Whenever such rights are stated and recorded in the SpecialRights table, the basic *TAM* computed results (CanView, CanEdit, CanDelete) are overridden with the SpecialRights stored values. The granular established special rights automatically take precedence over the basic rights evaluation mechanism.

In-dex	User Unique ID	Object Unique ID	Can View	Can Edit	Can Delete	Can Disable	Can Lock
int	bigint	bigint	bool	bool	bool	bool	bool

Fig. 6 Special Rights table layout

The mechanism of special rights is intended in case if there is a need to let a user manage objects above his role level, without raising the users rights or lower the objects requirements. A similar situation is if we need to forbid an user to access one more object that would normally have been in his administrative role range. The special rights are also non-inheritable. Here we make the observation that the TAM mechanism must not be obligatory suspended through object locking in order to make use of the SpecialRights feature. The SpecialRights feature is only an extension of the TAM functionality in order to permit special usage scenarios.

V. Conclusions

Fulfilling the security requirements of a Smart Building Controller, a secured access pattern was described in this paper in order to control, through an extensible hierarchy of 256 administration authority levels, the three important operations on SBC resources: read, write and delete, in increasing order of importance. These three authority levels implement the TAM - Three Authority Management pattern.

The TAM versatility comes from implementing the Special Rights table which maximizes the capability of granular administration of the SBC resources. By disabling the Three Authority Management strategy (through "Locking" the object) and enabling the Special Rights for one or more users, a clear specified, isolated and restricted environment management can be setup if required. The SpecialRights feature is only an extension of the TAM functionality in order to permit some special usage scenarios.

The main advantages of this resources access model are:

- Simple concept, easy to understand and use
- High granular access, limited only by what is defined as an object
- Can be added regardless of the secured resources model
- Easy to implement with minimal overhead
- Is thought as an additional security layer, modular
- Permits very flexible administrative scenarios

References

[1] "Basics of BACnet", http://kargs.net, 2014.
[2] ANSI/ASHRAE STANDARD Addendum 135-2001, "BACnet ® — A Data Communication Protocol for Building Automation," 2004.
[3] Contemporary Control Systems Inc., "BAS automation - Building on BACnet," 2013.
[4] Z. W. Z. Wang, X. L. X. Liu, and S. W. S. Wu, *BACnet intelligent home supervisory control system based on multi-agent*, vol. 2. 2005, pp. 761–764.
[5] W. Kastner, G. Neugschwandtner, S. Soucek, and H. M. Newman, "Communication Systems for Building Automation and Control," vol. 93, no. 6, 2005.
[6] R. H. Weber, "Internet of Things – New security and privacy challenges," *Comput. Law Secur. Rev.*, vol. 26, no. 1, pp. 23–30, Jan. 2010.

[7] R. Ausanka-Cures, "Methods for access control: advances and limitations," *Harvey Mudd Coll.*, 2001.
[8] E. Lee, "Cyber Physical Systems: Design Challenges," *2008 11th IEEE Int. Symp. Object Component-Oriented Real-Time Distrib. Comput.*, pp. 363–369, May 2008.
[9] D. Basin, M. Clavel, J. Doser, and M. Egea, "Automated analysis of security-design models," *Inf. Softw. Technol.*, vol. 51, no. 5, pp. 815–831, May 2009.
[10] S. D. Gribble, "Robustness in complex systems," *Proc. Eighth Work. Hot Top. Oper. Syst.*, pp. 21–26.
[11] D. Ferraiolo and D. Kuhn, "Role-based access controls," *Natl. Comput. Secur. Conf.*, no. 15, pp. 554–563, 1992.
[12] R. S. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST Model for Role-Based Access Control: Towards A Unified Standard," in *5th ACM Workshop on Role Based Access Control*, 2012, pp. 47–63.
[13] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," *IEEE Comput.*, vol. 29, no. 2, pp. 38–47, 1996.
[14] M. Nyanchama and S. Osborn, "Access Rights Administration in Role-Based Security Systems," *DBSec*, pp. 1–23, 1994.
[15] S. Osborn, R. Sandhu, and Q. Munawer, "Configuring role-based access control to enforce mandatory and discretionary access control policies," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 2, pp. 85–106, May 2000.
[16] M. Nyanchama and S. Osborn, "Modeling Mandatory Access Control in Role-Based Security Systems," *DBSec*, no. 1990, 1995.
[17] D. R. Kuhn, E. J. Coyne, and T. R. Weil, "Adding Attributes to Role-Based Access Control," *Computer (Long. Beach. Calif).*, vol. 43, no. 6, pp. 79–81, Jun. 2010.
[18] D. Kuhn, "Vulnerability hierarchies in access control configurations," *Safe Config, IEEE*, 2011.
[19] G. Stoneburner, C. Hayden, and A. Feringa, "Engineering Principles for Information Technology Security (A Baseline for Achieving Security), Revision A", 2004.
[20] K. M. Khan and J. Han, "Assessing security properties of software components: a software engineer's perspective," *Aust. Softw. Eng. Conf. ASWEC06*, p. 10 pp.–210, 2006.
[21] H. A. Weber, "Role-Based Access Control: The NIST Solution," *InfoSec Read. Room, SANS Inst.*, 2003.
[22] N. Kern, C. Kesavan, and A. Daswani, "Foundations of Security," *Foundations of Security*. Apress, pp. 3–24, 2007.
[23] A. Josang, B. AlFayyadh, T. Grandison, M. AlZomai, and J. McNamara, *Security Usability Principles for Vulnerability Analysis and Risk Assessment*, no. December. Ieee, 2007, pp. 269–278.
[24] D. R. Raymond and S. F. Midkiff, *Denial-of-Service in Wireless Sensor Networks: Attacks and Defenses*, vol. 7, no. 1. IEEE, 2008, pp. 74–81.
[25] L. Meyer and W. T. Penzhorn, *Denial of service and distributed denial of service-today and tomorrow*, vol. 2. 2004.
[26] R. K. Guha, Z. Furqan, and S. Muhammad, *Discovering Man-in-the-Middle Attacks in Authentication Protocols*. Ieee, 2007, pp. 1–7.
[27] B. Aziz and G. Hamilton, *Detecting Man-in-the-Middle Attacks by Precise Timing*, vol. 0. Ieee, 2009, pp. 81–86.
[28] A. M. Hagalisletto, *Errors in Attacks on Authentication Protocols*. 2007, pp. 223 –229.
[29] P. R. Babu, D. L. Bhaskari, and C. Satyanarayana, "A Comprehensive Analysis of Spoofing," *Int. J. Adv. Comput. Sci. Appl.*, vol. 1, no. 6, pp. 157–162, 2010.
[30] R. Weber and R. Weber, *Internet of things: legal perspectives*. Springer-Verlag Berlin Heidelberg, 2010.