Kang *et al. Hum. Cent. Comput. Inf. Sci.* (2017) 7:6
DOI 10.1186/s13673-017-0087-4

Human-centric Computing and Information Sciences

RESEARCH Open Access

An enhanced security framework for home appliances in smart home

Won Min Kang, Seo Yeon Moon and Jong Hyuk Park*

*Correspondence: jhpark1@seoultech.ac.kr
Department of Computer Science and Engineering, Seoul National University of Science and Technology (SeoulTech), 232 Gongneung-ro, Nowon-gu, Seoul 01811, Korea

Abstract

Since the end of 2000, smartphones have explosively spread and have made people's lives plentiful. With the start of smartphones, new smart devices, including tablet PCs, smart TVs, smart refrigerators, and smart air conditioners, have emerged, expanding their areas from individuals to business and home. These days, smart home service has drawn a lot of attention as a human-centric service. It is the environment where home appliances and other smart devices are connected to internet in order for user service and experience. The current smart home service is simply based on wireless home network to execute the function of connected home. The service has no smart home security so that it is possible for users to suffer a financial loss caused by information leakage or home appliance hacking. Therefore, to apply smart home service to surroundings, it is necessary to take into account security of smart devices. In this paper, we propose an enhanced security framework for smart devices in a smart home environment. The security framework provides the integrity system using the self-signing and access control techniques for preventing the security threats such as data modification, leakage and code fabrication.

Keywords: Smart device, Self-signing system, Data integrity, Home appliance, Access control

Background

With the development of internet of things (IoT), smart home appliances have been released. For instance, all home appliances for living, such as smart TVs, smart refrigerators, smart washers, and smart cooling and heating devices, have come to get connected to internet and make people's lives convenient. These days, the smart home service capable of integrating and managing the devices has globally developed. The smart home service is like a personal household based on automation. It has the system of controlling the internal and external factors of a house, such as lighting, temperature, doors, and windows. The smart home service is able to set lighting, temperature, music, and TV programs differently depending on whether a person is at home and a user's experience and preference [1–3].

These days, with the use of smartphones, it is possible to check home conditions, operate cooling and heating devices through temperature and humidity sensors, and control home lighting and TV to look as if a house is not empty [4, 5]. Also, in interoperation with smart devices, the smart home service offers living convenience in consideration of a user's living pattern and its accumulated experience. In other words, a user is able to control and monitor all home places with smartphone [6–8]. As such, the smart home service makes it possible for users to control their house regardless of time and space through smartphone. Although it is convenient and controls all home appliances, it has factors exposed to security threats, financial and psychological. Since smartphones supported various activities, such as financial affairs, business affairs, and personal affairs, the security technology of smartphones has been developed. However, the smart home appliances for smart home fail to have security technology and get exposed to many attacks [7, 9, 10]. Therefore, this paper proposes the internal security framework for smart devices, which can be applied to smart home appliances as well as smartphones. The proposed security framework uses self-signing technology to have defense against other infiltrating codes and executable programs and block execution flow. In addition, we explain the functions of modules which are limited by access control to protect the modules of home appliances in smart home.

This thesis is comprised of as follows: "Related works" section introduces the technology and research trend of smart home security and describes matters to be considered for security; "The proposed framework" section proposes an internal security framework for smart devices; the "Discussion" section describes a comparative analysis with related works; "Conclusions" section presents the conclusion of this study.

Related works

Since home appliances have wireless network function built in, smart home has provided a lot of services for users. Through wireless network, a user is able to control home appliances, lighting, and cooling and heating devices and receive services regardless of time and space. Smart home provides more convenient and useful services, for all home appliances get automated and smart. For useful services, a diversity of sensor information, customized personal information (hobbies, habits, medical service, etc.), and financial information are used. For the reason, security technology should be applied [7, 9–11]. This section describes the structure and security matters of smart devices in smart home, and the previous studies related to smart home security.

Architecture of smart device

By investigating such related works as [12–16], these researchers drew the structure of smart devices which can be applied to smart home. The structure of smart devices consists of four layers: application layer, application framework, module core library, and Linux kernel layer.

In application layer, applications running in smart devices are defined. In a smart home environment, there are a diversity of applications, such as games, launcher, healthcare application, set up, and video. These applications are not unique functions of smart devices, but they are developed by developers and third-party, running in smart devices.

In application framework, libraries, manager, and others to enable applications to run in smart devices are defined. The manager includes activity manager, package manager, and install manager, and manages libraries and files in execution.

Module core library processes the unique functions of smart devices. Each module in a smart device has its own unique function. For example, a smart TV has such modules as TV middleware for receiving images and displaying them on screen, player for playing images, TV configuration, and I/O kit. The modules execute each unique function of a device and can be inserted only by a relevant manufacturer.

Linux kernel layer executes the most core function of a smart device. It includes file system, network, kernel and device driver, and manufacturers change Linux kernel for each device function. In addition, it monitors the functional execution of a module and application, and exits or reruns execution.

Security requirements for smart home

In a smart home environment, services are provided over wireless network. An attacker is able to access or invade a smart device on the wireless network.

In this study, the security requirements for safe smart home service, including integrity, availability, and authentication, are discussed.

Integrity

A smart device can be accessed over wireless network, so that it needs security system. An attacker is able to insert a malignant software application and change a service purpose through a malicious code. For the reason, without integrity, the whole smart home system can be infected with a malicious code by an attacker and thereby the availability of smart home service can fall. Therefore, the integrity of smart home service is required. To ensure the integrity of smart devices, it is essential to use a hash function and a digital signature for critical data or module codes [16–18].

Availability

A smart device sends and receives data to and from the outside over wireless network. If an attacker steals data, it is possible to fabricate and modify the data. The fabricated data can cause malfunction of a smart device which deteriorates a user's availability of smart device. The deteriorated availability can lead to device overload that triggers a fire, and the malfunction can bring about financial losses like a rise in electric rate and the risk of life. To secure availability, it is required to limit other actions than essential functions and grant rights for functional access by making strong access control [7, 11, 16, 19].

Authentication

There are many devices whose security is not taken into account. If an attacker insert a copied module or a malignant code in a smart device, it is possible to contaminate a smart home service environment and make the device used for malicious purposes, such as distributed denial of service (DDoS), denial of service (DoS), and personal information leakage. Moreover, if an attacker disguises a modified module as a normal module, the module can serve as the secret backdoor for malicious action which can lower the function of the normal module and thereby deteriorate availability. Therefore, it is required to provide authentication of a smart device. For the authentication, it is possible to use a certificate [18–20].

Existing studies of smart home security

In this section, the previous works related to the service and security in smart home are looked into.

The firmware validation and update scheme proposed by Choi et al. [12] performs ID-based authentication between devices in a smart home environment and uses the key derivation algorithm for firmware image distribution. To verify the integrity of firmware image, it uses a hash chain. Firmware image is used as an input of the hash chain and is fragmented. The scheme transmits the pieces fragmented by firmware fragmentation, and put the transmitted pieces to the hash chain for verification.

The user privacy-enhanced security architecture proposed by Lee et al. [14] is applied in a smart home environment. The architecture has defense against such attacks as personal information hijacking and burst attack between an attacker and devices in a smart home environment. The study proposed a security framework applicable to a smart home environment, which includes encryption, access control, digital signature, authentication, and logging.

The framework proposed by Tomanek et al. [21] is based on the open source framework 'AllJoyn'. It is comprised of device, AllJoyn Core, permission module, and ACLs and policy certificate trust anchor. In the framework, critical data are transmitted after authentication between devices. End-user's security manager provides security provisioning and maintenance service for devices. A session is established between the applications of devices for data transmission. Authentication is performed with the use of a group key and a certificate. Authenticated devices transmit the messages encrypted with a given policy.

The lightweight lattice-based homomorphic privacy-preserving aggregation scheme proposed by Abdallah et al. [22] uses the authentication process of smart devices and lightweight lattice-based homomorphic cryptosystem to encrypt a message. It is divided into initialization phase and reading aggregation phase. Since the scheme makes it possible to monitor authentication between smart devices, control center, smart meter, and communication between APs, the control center is able to decrypt an encrypted message to improve confidentiality and privacy of devices.

To authenticate smartphones and send messages safely in a smart home environment, Mantoro et al. [23] uses an encryption algorithm and a hash function. The algorithm applies AES256, ephemeral Diffie-Hellman key exchange, and RC4-based hash function. With the use of a central hub, all messages to transmit are monitored, and the messages sent by smartphones pass the central hub for transmission. A message to transmit is encrypted with three algorithms, and a hash value is generated.

The proposed framework

In this section, the proposed framework is discussed in detail. The framework is comprised of module level and kernel level. The architecture of the framework is illustrated in Fig. 1.

Architecture

A smart device basically consists of modules which run the functions dependent on each device, and kernel which monitors and controls modules and module operation. It does

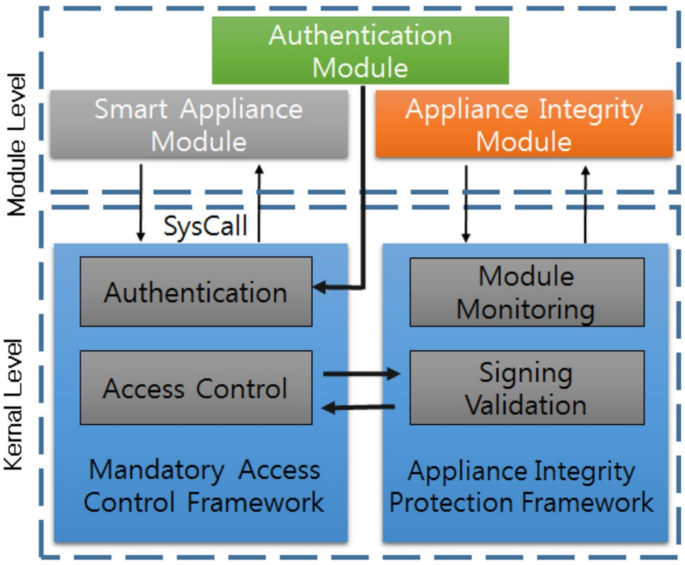

Fig. 1 Security framework inside the home appliances

not influence a running module and monitors an attack action. This framework is made up of smart appliance module, appliance integrity module, mandatory access control framework, and appliance integrity protection framework.

Smart appliance module is a functional module running in a smart device. It executes a basic function, rather than a particular function.

Appliance integrity module monitors all modules of a smart device. The modules of a smart device can run only when they have a manufacturer's signature. The modules without the signature exit by force.

Mandatory access control framework executes access control for all running modules. Each module has its own execution right and role. If a module executes an operation irrelevant to its role or has a different right, the module operation is limited by a system call.

Appliance integrity protection framework plays a role to examine actual self-signing. It has a list of signed modules and hash values saved. By comparing a hash value and a module name in the saved list, it checks whether a module has a manufacturer's signature. If a hash value and a module name are not found in the saved list, the module is classified as a newly added module and its signature is checked again.

Module level

In this section, module level applicable to a smart device is described. Module level is comprised of authentication module, smart appliance module, and appliance integrity module.

Authentication module

In a smart home environment, smart devices transmit data over wireless network. A variety of data are exchanged between devices. Such data can leak if an unauthorized device accesses the smart home environment. All smart devices in the environment perform authentication process through authentication module.

Smart appliance module

This module is dependent on a smart device. Each smart device has its own unique function. Diverse home appliances, such as refrigerators, cooling & heating devices, boilers, TVs, and lighting apparatus, perform their functions for user convenience. This module is used to express the unique functions abstractly.

Appliance integrity module

This module runs in the same level of mart appliance module. It monitors other newly created or inserted modules than the modules of manufacturers and validates integrity. Figure 2 shows check for signature flag function.

Kernel level

All functions of smart devices are controlled by embedded kernel. For integrity validation, the core framework is also executed in kernel level which is made up of mandatory access control framework and appliance integrity protection framework.

Mandatory access control framework

To execute a newly added module in a smart device, it is required to define module authentication and proper execution rights and roles of modules.

Authentication

A newly added module or an existing module installed by a manufacturer should be authenticated. Authentication module performs the module authentication process. The authentication module in the framework authenticates all modules when a smart device is powered on. The authentication process is pairing process.

Pairing process

In the authentication process, a certificate is created, issued, and verified with a manufacturer's private key. The issued certificate has a variety of information on a smart device,

```
    Sub CheckforSignatureFlag( )
      existing_Module =
CheckforModule(Module_Name)
      If existing_Module = True Then
        If SignatureFlag is "Signed" Then
            Print "Success"
          Call_Appliance Integrity Protection
Framework:
          SubFunction( )
        else If SignatureFlag is "Non-Signed" Then
            RequestKillModule(Module_Name)
      End If
      If existing_Module = False Then
        Call_Appliance Integrity Protection Framework:
        SubFunction( )
      End If
    END Sub
```

Fig. 2 Pseudo code of check for signature flag function

including a hash value of module ID, device name, device version, manufacturer's public key ID, issuer, and issuing time. Whether a module is already installed or newly added is checked, and each module requests Authentication Module for authentication. At the same time of request, a certificate is transmitted. Table 1 shows the terms explanation.

The following case shows that an existing module gives an authentication request to authentication module.

$$\text{Deliver } \mathrm{M_E} \rightarrow \ \mathrm{AM}: \ \text{Cert } \mathrm{M_E} = \left[\mathrm{Hash(MID),\ C\_TS,\ MN,\ MV,\ MF,\ K_{MF\_pub}ID}\right] \mathrm{K_{MF\_prv}}$$

Authentication module already saves a manufacturer's public key which is used to check a relevant module's certificate. A manufacturer's public is searched for with MF, $\mathrm{K_{MF\_pub}}$ and ID.

$$\text{Deliver } \mathrm{M_N} \rightarrow \ \mathrm{AM}: \ \text{Cert } \mathrm{M_N} = \left[\mathrm{Hash(MID),\ C\_TS,\ MN,\ MV,\ MF,\ K_{MF\_pub}ID}\right] \mathrm{K_{MF\_prv}}$$

In the case of a newly added module, authentication module doesn't have the module manufacturer's public key. Therefore, authentication module requests certificate authority to send a public key and save the key newly.

$$\text{Return AM} \ \rightarrow \ \mathrm{M_E} \text{ or } \mathrm{M_N}: \ \text{Result} = [\mathrm{Hash(MID),\ CF,\ CF\_TS,\ MV,\ MN}]\ \mathrm{K_{MF\_pub}}$$

Once pairing process is complete, authentication module returns result values to a relevant module. The results values are module ID, the creation time of the results value, module name, module version, and checked flag. They are encrypted with a manufacturer's public key before being transmitted. If authentication successes, checked flag has "Permitted"; if authentication fails, it has "not permitted". Authentication module saves result values. After that, if the same module is inserted, the module checks checked flag of the result values, and, if "not Permitted", it makes an authentication request again.

Access control

The header of each module has its information section where there is module information. In the information section, module name, version, permission, identifier, installation path, role, and other information are saved. Figure 3 presents the information section of the header in a module.

Table 1 Terms explanation

Terms	Explanation
$\mathrm{M_E/M_N/AM}$	Existing/new module/authentication module
MF/CA	Manufacturer/certificate authority
Hash ()	Hash function
$\mathrm{Sig_x()/Very_y()}$	Signature algorithm using x key/ Verification algorithm using y key
Cert	Certificate
$\mathrm{K_{x\_prv}/K_{x\_pub}}$	Private key of X/public key of X
MID/MN/MV	Module identifier/module name/module version
X_TS	Time stamp values of X
CF	Checked flag

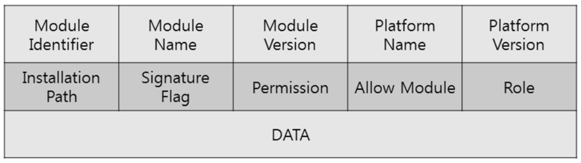

Fig. 3 Information section of the header in the module

Permission specifies the right of module execution. Allow module is the identifier of a module that can access information on the current module. The detailed roles are presented in Table 2.

A role is divided into name and value. Each name has the value of Boolean, String, or Integer. Such a role should be specified by a manufacturer and developer before a module is executed. Once the module is executed, it is impossible to change and edit the role.

In reference to the information section, a list for access control is created. The list is used for module monitoring and signing validation of appliance integrity protection framework. A list is generated on the basis of module information and roles which are used to make only essential functions of a module executed. The information section of module header has no 'write' permission in order for an attacker not to change a role at its discretion. Table 3 displays an example of role reference list.

A list created by access control consists of module name, permission, role, and value. For instance, the execution permission of a camera module should be user permission in which it is possible to access a photo and run as the frontend module of a smart device. RecognizeVoice module should be operated with user permission in which it is possible to call an installed application but impossible to write a character and access the passwords stored.

Appliance integrity protection framework

All modules in a smart should have integrity. Although an attacker makes a fabrication, it is possible to endure integrity through verification. appliance integrity protection framework is divided into module monitoring and signing validation.

Module monitoring

Module monitoring monitors an existing module and a newly added module. Running together with appliance integrity module, it delivers information on a newly added module to singing validation so as to check integrity. Figure 4 presents Pseudo Code of ModuleMonitoring Function.

The called ModuleMonitoring function checks the integrity of an operating module or a module in ready. CheckforModule function checks if the module exists already. LoadRoleTable function loads the role list of a relevant module saved in Mandatory Access

Table 2 Examples of role list

Role	Value	Description
Allow-attach-debugger	Boolean	Debugger can be attached to module
Allow-save-to-X	Boolean	Output can be saved in the storage X
Allow-open-application	Boolean	In module, an application can be opened
Allow-perform-background	Boolean	Module can run in the background
Create-child-process	Integer	The number of child processes that a module can create

Table 3 Examples of role reference list

Pointer num	Module name	Permission	Role	Value
R1-0	Camera	User	Allow-save-to-photo	True
R1-1	Camera	User	Allow-open-to-frontend	True
R2-0	RecognizeVoice	User	Allow-open-application	Ture
R2-1	RecognizeVoice	User	Allow-insert-to-string	False
R2-2	RecognizeVoice	User	Allow-access-passwdstore	False

```
   Sub ModuleMonitoring( )
      Do Forever
      existing_Module =
CheckforModule(Module_Name)
      MAC_List = LoadRoleTable(Module_Name)
      Deny_Role_List = "Allow-attach-debugger"
      Module_List = ExtractRole(Module_Name)
      If existing_Module = True Then
        If MAC_List = Module_List Then
            Print "GOOD"
          #performing the module
          SubFunction( )
        else
            RequestKillModule(Module_Name)
      End If
      If Module_List = Deny_Role_List Then
        RequestKillModule(Module_Name)
      End If
   END Sub
```

Fig. 4 Pseudo code of ModuleMonitoring function

Control Framework. The role of the module to check is extracted by ExtractRole function, and its results values MAC_List and Module_List are compared for integrity check. If the items of the list are different, the module exits by force. In addition, if the role that the module should not perform is given to Deny_Role_List and thereby the module tries to run the role in the list, it is required to prevent the execution of the module and exit the role by force if the role tries to be performed in the middle of running module.

Signing validation

the second phase of integrity verification is to test a code at the time when a module starts. Each module has the certificate signed by its manufacturer. It is checked that a module is signed already. Signing validation uses a hash function to check all codes of the module. In the process, the secondary integrity is verified. In this phase, whether there is any modification or fabrication of an attacker is checked.

For integrity verification, appliance integrity protection framework saves a list of hash values for module codes and is able to reject the execution of a module which is not matched in the saved hash list after mapping. Table 4 shows examples of hash values list.

A hash list is made up of module name, hash values, role table pointer, and pointer number. The hash values of a module are compared first, and then role matching is performed again. For role matching, the role reference list created by mandatory access control framework is referred to. With the use of the pointer number of the role reference

Table 4 Examples of hash values list

Module name	Hash values	Role table pointer	Pointer num
Camera	c1ae36d8def8c4242710b4e8544e956c	R1	1
RecognizeVoice	b1d905f82b6f4c7ab4dc66ed40b896f8	R2	2
StorageMounter	0a2f90d128261d77b2af01342a979971	R3	0
CrashManager	4b9e6227c0117cbf301946eb1a554307	R4	5

Table 5 Comparison analysis

Classification	Byung-Chul Choi et al. [12]	Ondrej Tomanek et al. [21]	Asmaa R. Abdallah et al. [22]	Proposed framework
Integrity	△	△	△	O
Availability	X	X	△	△
Authentication	O	O	X	O

O, strong; △, medium; X, weak

list, a list value is saved and compared. This process is performed before the module is executed. Therefore, it is possible to make the secondary integrity verification.

Discussion

In this section, the proposed frame work is compared with relevant works in terms of the security factors in a smart home environment: authentication, integrity, and availability. We explain the comparison analysis in Table 5.

Integrity

All data transmitted by smart devices should ensure integrity. Moreover, all modules inserted in a smart device should keep integrity. In a smart home environment, a variety of information is shared by devices, and different types of modules can be inserted in a device by a developer or a user. Therefore, it is necessary to provide integrity of modules, monitor normal activities of modules, and check any modification. The framework proposed in this study checks overall codes of a module through self-signing. Before the insertion into a device, verification is performed in reference to a hash list. Each module has its own roles. If a module tries to play a role not specified, module monitoring exits the module by force or makes it turn to ready status and restart the module. In the two processes, the operation of module codes is monitored so as to ensure integrity. In the case of [12], encryption and hash chain are applied to all data transmitted by manager and server in order to provide integrity. A device checks all fragmented hash values and use the verified data. In the case of [21], integrity is provided through encryption. In [22], the range of integrity is limited. Consumption's messages are encrypted before transmission, so that they are safe against Man-in-the-middle attack. However, in [12, 21, 22], integrity is not ensured for the messages or operations of consumption or other appliance modules which are already authorized and run normally.

Availability

Availability is the most important factor of all smart devices in a smart home environment. Lowering availability can lead to the deterioration of the original function of a smart device, which can cause user inconvenience, financial loss, and psychological damage. To prevent availability deterioration, the proposed mandatory access control framework reads the header of a module and creates and saves a role list. By specifying the roles of a module, it is possible to prevent any malicious actions other than the specified roles. The created list is loaded by module monitoring of appliance integrity protection framework, and the information section of the module and the list are compared before module execution so as to judge the roles of a module and its execution permission. By addition a role of a module, it is possible to alleviate availability deterioration. In [12, 21, 22], availability evaluation is not taken into account. In [21], the limits of AllJoyn are described. Module transformation can cause optimization and deadlock, so that availability can lower. According to the analysis on computation overhead in [22], the proposed scheme is able to simplify the vector space used for encryption, and thus there is no influence on performance. However, if a malignant module lowers availability, the result will be different.

Authentication

All modules inserted into a smart device should be authorized. Authentication process is applied to not only existing modules, but newly added modules. It is performed in pairing process. When an existing module or a newly added module sends a certificate to authentication module in module level, the authentication module receives a return value for authentication from the authentication of mandatory access control framework in kernel level. The return value is encrypted first and then transmitted to a relevant module. In the pairing process, the execution permission of a module is checked. If "not Permitted", the module is not permitted to be executed. In [12], the firmware image is transmitted, and an ID is offered differently depending on devices and is encrypted before transmission. With the use of the ID, a device is authenticated. In [21], security manager issues a certificate-typed security provisioning. Provisioning includes information on security group and certificate information membership. Through provisioning, authentication is processed. In [22], there is no description of the authentication for modules or devices.

Conclusions

As smart home service has gradually been provided for users, the issue of security threat has emerged. Most smart devices in a smart home environment fail to ensure security, so that they are exposed to many threats. Therefore, this study proposed an internal security framework for smart devices in smart home in consideration of data integrity, invasion of non-permitted devices, and availability. The proposed framework provides the security service for ensuring device authentication, integrity, and availability. The framework can have a limited range, for it needs to offer security service in the range where the unique functions of a smart device are deteriorated. In the future, it will be necessary to propose a more optimized security framework for smart devices, make it lightweight, and maximize its availability.

Authors' contributions

WMK and SYM carried out related works and design of the proposed scheme. JHP mainly managed and supervised this paper. All authors read and approved the final manuscript.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No 2016R1A2B4011069).

Competing interests

The authors declare that they have no competing interests.

Received: 6 December 2016 Accepted: 20 January 2017
Published online: 05 March 2017

References

1. Kim MH, Shin YT (2016) A study on the smart home service security threat. Proceedings of symposium of Korean institute of communications and information sciences, pp 1069–1070
2. Ryu HS, Kwak J (2015) Group key management method for secure device in smart home environment. J Korea Inst Inf Secur Cryptol 25(2):479–487
3. Lee JG, Yang CS, Kim JH, Kim KJ (2015) A research and development of integrated platform for data security between different smart home devices. J Korea Inst Inf Commun Eng 19(5):1173–1179
4. Lévy-Bencheton C, Darra E, Tétu E, Dufay G, Alattar M (2015) Security and resilience of smart home environments. European union agency for network and information security
5. Rajiv P, Raj R, Chandra M (2016) Email based remote access and surveillance system for smart home infrastructure. Perspect Sci 8(2016):459–461
6. Fernandes E, Jung J, Prakash A (2016) Security analysis of emerging smart home applications. 2016 IEEE symposium on security and privacy, pp 636–654
7. Jacobsson A, Boldt M, Carlsson B (2016) A risk analysis of a smart home automation system. Future Gener Comput Syst 56(2016):719–733
8. Markantonakis K, Akram RN, Msgna MG (2016) Secure and trusted application execution on embedded devices. Innovative security solutions for information technology and communications, vol 9522, pp 3–24
9. Kumar P, Gurtov A, Iinatti J, Ylianttila M, Sain M (2016) Lightweight and secure session-key establishment scheme in smart home environments. IEEE Sens J 16(1):254–264
10. Madakam S, Date H (2016) Security mechanisms for connectivity of smart devices in the internet of things. Connectivity frameworks for smart devices: part of the series computer communications and networks, pp 23–41
11. Jose AC, Malekian R (2015) Smart home automation security: a literature review. Smart Comput Rev 5(4):269–285
12. Choi BC, Lee SH, Na JC, Lee JH (2016) Secure firmware validation and update for consumer devices in home networking. IEEE Trans Consum Electron 62(1):39–44
13. Hasan Hussain S, Geetha S, AmuthaPrabhakar M (2016) Design and implementation of adaptive model for sustainable home automation using internet of things (IoT). Int J Adv Eng Technol 7(1):827–829
14. Lee S, Kim J, Shon T (2016) User privacy-enhanced security architecture for home area network of smartgrid. Multimed Tools Appl 75(2016):12749–12764
15. Lin H, Bergmann NW (2016) IoT privacy and security challenges for smart home environments. Information 7(3):1–15
16. Lee C, Zappaterra L, Choi K, Choi HA (2014) Securing smart home: technologies, security challenges, and security requirements. Workshop on security and privacy in machine-to-machine communications (M2MSec'14), pp 67–72
17. Komninos N, Philippou E, Pitsillides A (2014) Survey in smart grid and smart home security: issues, challenges and countermeasures. IEEE Commun Surv Tutor 16(4):1933–1954
18. Schiefer M (2015) Smart home definition and security threats. 2015 ninth international conference on IT security incident management & IT forensics, pp 114–118
19. Agosta G, Antonini A, Barenghi A, Galeri D, Pelosi G (2015) Cyber-security analysis and evaluation for smart home management solutions. The 49th annual IEEE international carnahan conference on security technology, pp 1–6
20. Chitnis S, Deshpande N, Shaligram A (2016) An investigative study for smart home security: issues, challenges and countermeasures. Wirel Sens Netw 8:61–68
21. Tomanek O, Kencl L (2016) Security and privacy of using AllJoyn IoT framework at home and beyond. 2016 2nd international conference on intelligent green building and smart grid (IGBSG), pp 1–6
22. Abdallah AR, Shen XS (2014) Lightweight lattice-based homomorphic privacy-preserving aggregation scheme for home area networks. 2014 sixth international conference on wireless communications and signal processing (WCSP), pp 1–6
23. Mantoro T, Ayu MA, binti Mahmod SM (2014) Securing the authentication and message integrity for smart home using smart phone. 2014 international conference on multimedia computing and systems (ICMCS), pp 1–5