Security Concerns and Countermeasures in IoT-Integrated Smart Buildings

Kruthika Rathinavel, *Student Member, IEEE*, Manisa Pipattanasomporn, *Senior Member*, *IEEE*, Murat Kuzlu, *Senior Member*, *IEEE*, and Saifur Rahman, *Fellow*, *IEEE*

***Abstract*— The integration of Internet-of-Thing (IoT) devices to smart buildings raises the risk of vulnerabilities in building operation. This paper presents the software development work for the secure deployment of IoT devices in commercial buildings. Security threats with their countermeasures are analyzed in the context of a specific Building Automation System (BAS) implementation called Building Energy Management Open Source Software (BEMOSS™).**

***Index Terms*— Building energy management, Internet of Things (IoT) and IT security.**

I. Introduction

Building Automation Systems (BAS) provide the ability to monitor and control building operation remotely. These systems are generally deployed to monitor and control major loads, such as Heating, Ventilation and Air Conditioning (HVAC), lighting and plug loads. BAS systems are beginning to deploy low-cost, open standard and open source IoT devices. With this shift, new security challenges have emerged. A BAS system with IoT devices is typically connected to the cloud, thus exposing the system to vulnerabilities at the IP level. Compromising a single device in the network can result in compromising the safety and security of the entire building.

In recent years, BAS systems have been exposed to several cyber attacks. For example, the Tridium Niagara–a commercial BAS system in Google's Australia office–was hacked [1]. In another instance, hackers gained control of building locks, electricity, elevators, etc. as reported in [2]. HP found several vulnerabilities in its recent security assessment of BAS [3]. Some of these attacks can be thwarted by securing networks and communication protocols. Related work [4, 5, 6] discusses about securing the network layer of a BAS platform, including a means to secure communications protocols, like LonWorks, Zigbee and BACnet. In [7], authors analyze the general security issues and provide coping strategies for safer IoT construction layers. Challenges in IoT security, including object identification, authentication and authorization, lightweight cryptosystems and security protocols, are discussed in [8]. Authors in [9] discuss a technique called named-data networking, which focuses on information-centric network architecture designs. In [10], authors enumerate some research directions for IoT, targeting issues at the application layer, e.g., user privacy protection, leakage of sensitive information, destruction of computer data, database corruption and other vulnerabilities. Authors in [11] indicate privacy and security are some of the most prominent challenges when integrating IoT devices to a network and IoT integration demands additional security which researchers and industries often overlook.

With the rapidly growing network of connected devices (i.e., IoT), it has become increasingly important to secure applications, such as a BAS system that controls IoT devices. While BAS systems should be secured end-to-end, privacy of building operational data is also important. While such details are enumerated everywhere, very few papers discuss ways to implement such security measures. Often times the application layer and user interaction layer security is not addressed in these discussions. Additionally, while web security has been discussed as a standalone concept, it has not been discussed in the context of a BAS system integrating IoT devices, which includes several additional layers as opposed to a traditional web application.

This paper discusses cybersecurity issues for IoT device integration in a smart building (focusing at the device level, platform level and the web interface) and provides countermeasures for threats and vulnerabilities in the context of a specific BAS implementation called BEMOSS™.

II. Overview of BEMOSS™ and its Architecture

BEMOSS™ (www.bemoss.org), sponsored by the U.S. Department of Energy at Virginia Tech, is an open source software platform built on a multi-agent technology. It has been designed to allow sensing and control of HVAC, lighting and plug loads in small- and medium-sized commercial buildings that do not have existing building automation systems. As an alternative to expensive proprietary BAS solutions, BEMOSS™ has proven to reduce energy consumption and enable implementation of demand response in buildings. BEMOSS™ also enables seamless interaction between users and IoT devices, such as Google Nest Thermostat, Philips Hue, WeMo smart plugs, NetAtmo environmental sensors, and many others.

A. BEMOSS™ Software Architecture

The BEMOSS™ software architecture consists of four layers, as illustrated in Fig. 1 [12]. These layers communicate with BEMOSS™ databases to manage metadata and time-series data. Each layer is discussed in details below.

This work was supported in part by the U.S. Department of Energy under Contract DE-EE 0006352.

978-1-5386-2890-4/17/$31.00 ©2017 IEEE

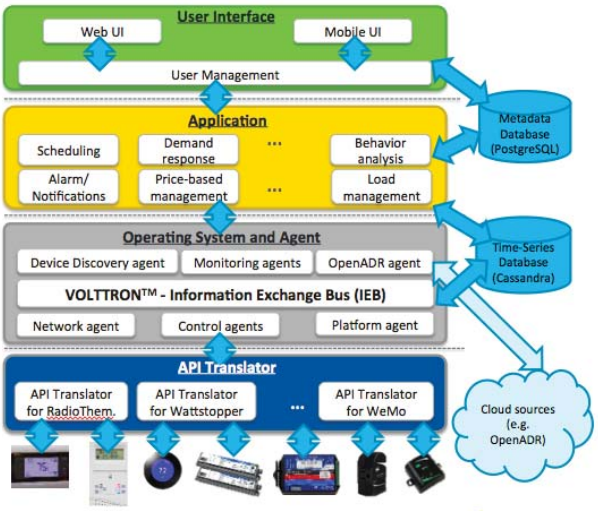

Fig. 1 BEMOSS™ software architecture.

UI Layer is responsible for provisioning the graphical user interface and user management. BEMOSS™ UI is a responsive web platform that allows seamless interaction with sensors and load controllers through a web browser. This layer has built-in authentication and authorization to allow different levels of access control and provides secure web access.

Application Layer allows implementation of various intelligent control applications for the platform. Some of the possible applications are: fault-detection and diagnostics, price-based building control and management, load shape analysis, demand response, planning and scheduling, behavior pattern analysis, load management, alarms and notifications.

Operating System and Agent Layer is considered the root of the application, where VOLTTRON™ and BEMOSS™ agents reside. Developed by the Pacific Northwest National Laboratory, VOLTTRON™ is an agent-based platform that provides low-level communications and necessary modules for high-level application development [14]. Utilizing the VOLTTRON™ platform to provide foundation for agent communications, BEMOSS™ has been developed. VOLTTRON™'s Information Exchange Bus (IEB) serves as a medium that enables communication among all BEMOSS™ agents. The IEB also enables communication with the UI. The UI layer connects to the IEB using a specific UI Subscriber/Publisher agent. Note that there is a one-to-one relationship between a BEMOSS™ agent and a device for monitoring and control.

API Translator Layer is responsible for communication between BEMOSS™ agents and all sensors, controllers and IoT devices. API translators have been developed to allow BEMOSS™ agents to communicate with a group of devices based on their unique communication protocols and Application Programming Interface (APIs). This allows BEMOSS™ to read and control connected devices regardless of their difference in API.

BEMOSS™ Database: In BEMOSS™, there are two types of databases, one to store metadata information about different devices (using a relational database) and the other to store time-series data.

B. BEMOSS™ System Architecture

When BEMOSS™ is deployed in a building, each BEMOSS™ node (defined as a single-board computer with BEMOSS™ source code installed) can communicate with sensors and controllers in a building via a wireless network. BEMOSS™ multi-node architecture can be deployed to support multi-floor/multi-zone buildings. In our previous work [12, 13], the BEMOSS™ system architecture is given in detailed.

III. SECURITY GOALS OF BEMOSS™

The security goals for BEMOSS™ are represented in the CIA (Confidentiality, Integrity, and Availability) triad [15]. CIA is a widely used benchmark for evaluation of information systems security, focusing on the three core goals of confidentiality, integrity, and availability of information.

Confidentiality is the inability of contents to be intercepted by unauthorized personnel. As BEMOSS™ relies on wireless networks for its communications, maintaining confidentiality in information exchange is critical. Communication between the client (web browser) and the server (application backend) and between different nodes in BEMOSS™ can be compromised if it is not adequately secured.

Integrity is the guarantee that data is protected from accidental or deliberate (malicious) modification. Integrity refers to the trustworthiness of information resources. If data communication between different layers of the application is not encrypted, it can be manipulated/intercepted in transit. Integrity for data in transit is typically provided by using hashing techniques and message authentication codes. BEMOSS™ is accessible over the Internet making it vulnerable to a host of attacks if the application is not properly secured.

Availability of information refers to ensuring that authorized parties are able to access the information when needed. The most common attack is Denial of Service (DoS) attacks. The primary aim of DoS attacks is to deny users of access to resources of a website. In a system like BEMOSS™, IEB can be bombarded with messages, causing the message queue to overflow and lose messages to be received and acted upon. Brute force login attacks and username enumeration can slow down/shut down the system if they are not dealt with adequately.

BEMOSS™ also provides failure recovery and redundancy in both system operation and data storage. This software platform deploys its multi-node architecture – that is engineered such that a failing node can transfer its responsibilities to the core – to enable continuous monitoring and control of building environmental functions. Building operation data stored in BEMOSS™ can also be distributed to multiple nodes for redundancy.

Additionally, an end-user facing web-application is also vulnerable to a host of vulnerabilities and other security threats. The OWASP (Open Web Application Security Project) [16] has identified top web application security risks. BAS applications usually communicate with sensors and controllers to enable monitor and control of building loads, using a communication interface – usually a RESTful service or a message queue

978-1-5386-2890-4/17/$31.00 ©2017 IEEE

depending on application requirements. This communication interface is also vulnerable to attack over the Internet.

IV. Security Implementation

To protect the application from the above threat scenarios, several security features are implemented in BEMOSS^{TM}, which are discussed below:

A. Preventing Brute Force Attacks

Brute force attacks (or an exhaustive key search method) is a cryptanalytic attack that can, in theory, be used against encrypted information. Unlike regular hacking that focuses on vulnerabilities in software, a brute force attack aims at being the simplest kind of method to gain access to a site by trying usernames, passwords, repeatedly, until the password is cracked. Although this is inelegant, when unsafe passwords are used, these attacks are easily possible. Due to the nature of these attacks, the server memory is often used up, which can cause degradation in performance of a server. In BEMOSS^{TM}, the "Enforce Strong Passwords" and "Lock After Failed Login Attempts" steps have been taken to prevent brute force attacks.

B. Secure Communication using Transport Layer Security

To allow no compromise of user credentials and sessions, an SSL encryption is necessary. Although the use of SSL slightly slows down the application, it is important to have a secure communication between various layers of the application. BEMOSS^{TM} implements an SSL encryption by acting as its own certificate authority (CA) to encrypt traffic end-to-end between client and server. Since BEMOSS^{TM} is mostly internal to a building, issuing its own certificates is a reasonable way to ensure security, being open source. BEMOSS^{TM} being its own CA needs to be registered with the browser the first time a user logs into the site. Connections between the client and server are encrypted using AES. The key size used for encryption is 256 bits. It operates on the Galois/Counter Mode (GCM) [17], which is used for authenticated encryption. Certificates and keys are added to the server and the server is configured to start up and function in the encrypted mode. The same function can also be performed with a verified certificate authority, like Verisign, GlobalSign, if required.

C. Web Sockets Security

Similar to encrypting HTTP over SSL/TLS, web sockets can also be encrypted using SSL/TLS. This protects the system against man-in-the-middle attacks. Tunneling any important requests to the underlying multi-agent platform has been avoided. All requests made by the user to control a device or modify settings happen using the web server request/response protocol thus avoiding websocket level requests. On the other hand, a push update from a specific agent to the application is handled using Tunneling requests like database connection can lead to an XSS attack thereby leading to the breach of the entire application. Client inputs through a web socket are completely avoided. All client inputs are validated at both the client side and the server side, and requests are redirected to the core platform via the server. No client requests are sent through web sockets. Thus, the application cannot be infected via a client side web socket request. The same-origin policy that rejects any requests or pings from cross-origin servers is also implemented to secure BEMOSS^{TM}.

D. Session Management

The session framework allows retrieval of necessary information based on the user that is logged in. It stores data on the server side and abstracts the sending and receiving of cookies. Cookies contain a session ID – not the data itself. Sessions are implemented using a session middleware that provides a database backed session. A session is stored as a data model in the metadata storage system. Each session is identified by a pseudo-random 32-character hash stored in a cookie. Decoding the session ID gives the user ID in the JSON format, which can be used to perform required operations. Using database-based cookies is safe since it defines the next level of indirection for an attacker trying to misuse the session. A cookie-backed session might be susceptible to replay attacks, since cookies are not freshness-guaranteed and do not usually have an expiration date. This means that an attacker who obtains access to a session ID while a user still logs in can store/copy cookies and access a user account even after a user logs out.

E. Cross Site Request Forgery (CSRF) Protection

A CSRF attack relies on the fact that the browser manages cookies, and will include cookies associated with a target domain to the forged HTTP request. The BEMOSS^{TM} user platform uses a Double Submit Cookie. This provides enough protection from a CSRF attack since it is impossible for an attacker to control the cookie field in a CSRF attack. A Double Submit cookie is defined as sending a random value in both a cookie and as a request parameter, with the server verifying the cookie value and the request value. A CSRF cookie is set in the HTML page using a 'hidden' field. This ensures that the server receives the CSRF token without any malicious code that modifies the token value. Custom JavaScript is written that acquires the CSRF token and sends it to the server for every AJAX (Asynchronous JavaScript and XML) request sent. A CSRF cookie is generated as a session independent nonce value. Other sites do not have access to this cookie. The safety of this technique also relies on how random the CSRF token is. The random number generator used in this context is the Mersenne Twister Algorithm [18]. This provides protection against identification spoofing, cookie replay attacks and CSRF attacks.

F. Cross Site Scripting (XSS) Protection

XSS is defined as one of the primary security loopholes in many web-based applications where user inputs are taken into the system unsanitized. When generating HTML from templates, there is always a risk of a malicious user entering harmful JavaScript and other content into a form. In BEMOSSTM, each untrusted variable is run through an escape filter that converts potentially harmful HTML characters to safe elements. In BEMOSS^{TM}, majority of user inputs are standardized. These standardized inputs have minimum use of text fields where users are required to type in a value. This way, most of the causes for XSS attack can be avoided. In other

978-1-5386-2890-4/17/$31.00 ©2017 IEEE

cases, the client-side and server-side verification of the user input is performed. The only area in the web application that involves user input validation is for nicknames (device nicknames, zone nicknames, schedule nicknames). Using standardized inputs and server-side validation provides protection against SQL injection attacks.

G. Error Handling

With web/general desktop applications, the error dump on the screen is detailed enough for a developer to examine the cause of the error. Such detailed information is a good development tool, but it reveals too much information to users. For this reason, it is important to abstract away error information from end users. BEMOSS™ abstracts away all error messages to secure away the source code and implementation details from end users or attackers. Once an error, like 404, 500, 302, etc., is captured in the server, and a user is redirected to the corresponding error page. For example, when a page that is not part of the regular URL configuration is accessed, it is redirected to a 404 error page abstracting the server process.

H. User Authentication and Authorization

Authentication: The first step in creating a secure web application for BAS access is to enable user authentication to the system. To access BEMOSS™, a user is redirected to the login screen where the user is authenticated. His privileges are also registered at this point that allows user experience customization based on his permissions. Authentication is based on username/password. The PBKDF2 algorithm with HMAC-SHA256 [19] hash with ~20,000 iterations is used to secure the password in the database.

Authorization: An authenticated user is authorized to access parts of the application or the entire application based on role (discussed next). Each node is configured with the authorization data. Each user has an Access Privilege Chart (APC), which helps determine the user's privilege at runtime. The APC can be dynamically adjusted by an administrator. BEMOSS™ is unique in restricting user privileges to the UI level access and denies a user any privilege to access the underlying multi-agent layer directly. Some security features implemented at the multi-agent layer also prevent the user from manually tampering with IoT devices by employing intelligent access restrictions. If a user's privileges are modified, the information is forwarded to the session and new privileges take effect immediately.

I. Role-based User Management

BEMOSS™ platform includes role-based user management to secure the system. Users are categorized as:

- Administrator – is given overall authorization to monitor and control all devices in building operations.
- Zone manager – is given authorization to control devices located in the zone assigned to him/her.
- Tenant – has read-only access to parts of the system.

Roles are associated with APC and thus ensure that every request sent to the web server is handled based on the user's role. Malicious requests are discarded.

J. IP blacklisting:

BEMOSS™ takes measures in the software implementation to prevent denial of service attacks. A middleware is implemented in the user platform layer to lock out or banish users by IP address. Users are automatically banned if they exceed a certain number of requests per minute, which is considered a likely form of denial of service attacks.

K. Securing IEB

A typical BEM system often exchanges messages using web services or message queues. In the BEMOSS™ implementation, a message queue is used. The message queue is implemented as a publish-subscribe (Pub-Sub) mechanism. Publishers are created with the PUB socket type. Data is published on a particular topic. A subscriber can choose to subscribe to these topics. Subscribers are created using the SUB socket type. A subscriber can connect with many publishers. Messages can be filtered based on topics. Note that the Pub-Sub communication is asynchronous. That is, if a 'publish' service has been started and a 'subscriber' subscribes to a topic already published, those messages would not be received by the subscriber. Only messages published after a subscription is enabled will be received. If the subscriber connects to more than one publisher, the data arrive interleaved, thus not allowing a single publisher to drown the queue with its messages. Filtering happens at the subscriber end, not at the publisher end.

There are two types of communication protocols used: inter process communication (IPC), which needs ipc endpoint names instead of an IPv4 address to be used, and transmission control protocol (TCP).

L. Device Approval Process

In BEMOSS™, it is important to add only legitimate devices to a building system. An attacker may add fake devices to the building operation, and try to attack the platform. For this reason, a device approval process is necessary. BEMOSS™ has an automatic device discovery process that runs in the background to discover new devices as they become available in the network. However, an approval is required from a building administrator before a device is added to BEMOSS™.

M. Isolating Sensitive Information

In an open source implementation where plenty of developers access a repository, check out and use the same code, it is important that sensitive information be protected and kept unique. Securing sensitive information by placing them in areas that cannot be reached can prevent any security flaws in the system. However, in case of a system compromise, most of this information can be leaked.

In BEMOSS™, there are several usages of a secret key, which is used for cryptographic signing within an application. It is important to safeguard these keys at all times. A secret key is used for the following functions in the application, directly or indirectly:

- JSON object signing
- Password reset token
- Form security
- Protect against message tampering
- Protect session data and create random session keys

978-1-5386-2890-4/17/$31.00 ©2017 IEEE

- Create random salt for most password hashers
- Create random passwords, as required
- Create CSRF tokens

N. *Time-series Database Security:*

BEMOSS™ web server reads the time-series database to provide live statistics about the various entities/devices that are part of the building automation system. The web server has been designed in such a way that it can only read from the database, both metadata and time-series. To write data to the database, the web server has to route it through the VOLTTRON™ platform and its agents. The read access to the database is controlled by an authentication method that secures the underlying databases from being tampered/accessed by the attackers.

O. *Linux Platform Security*

BEMOSS™ is developed in the Linux Ubuntu environment. For BEMOSS™, relevant security modules are constantly installed and implemented. System log utilities like systems [20] are installed to obtain system logs. The core platform and the user platform constantly add logs to files to allow monitoring of application activities and for debugging purposes. The 'Snort' intrusion detection tool is installed and a set of rules are implemented to monitor system intrusions.

P. *Device Security*

Since some devices that are accessible by BEMOSS™ have a basic HTTP protocol, using which an adversary will be able to control the device if the device IP address is obtained. In case of gaining such access, including the case of gaining physical access to the device, an attacker can control a thermostat by tampering with its temperature set point. BEMOSS™ tackles this issue by detecting such a set point change and revert the thermostat temperature set point back to its original value, if the thermostat override is not allowed by the administrator.

V. CONCLUSION

This paper presents cybersecurity issues in IoT device integration in smart buildings. It lays out possible security threats and vulnerabilities, as well as presents countermeasures in the context of a remote access BAS. It is important to note that one of the limitations with implementing a security framework that involves encrypting and decrypting data is the use of processing power. BEMOSS™ is targeted at low cost embedded systems, like the Raspberry Pi and Odroid type devices, which have lower processing power compared to traditional laptops or desktop computers. The BEMOSS™ security framework runs alongside the entire computing application/platform that controls these devices. It is important that the security framework does not consume a lot of processing power and provides optimum security for the entire system.

This paper has provided an insight into the security concerns in building-integrated IoT devices for software developers and researchers who work with open source software.

REFERENCES

[1] Infosecurity (Magazine) - Researchers hack Google's Australian office building [Online]. Available: http://www.infosecurity-magazine.com/news/researchers-hack-googles-australian-office. Retrieved: Oct 2015.

[2] Wired (Magazine) - Vulnerability Lets Hackers Control Building Locks, Electricity, Elevators and More [Online]. Available: http://www.wired.com/2013/02/tridium-niagara-zero-day. Retrieved: Oct 2015.

[3] The Register (Magazine) - Internet of Thieves: All that shiny home security gear is crap, warns HP [Online]. Available: http://www.theregister.co.uk/2015/02/10/iot_home_insecurity. Retrieved: Oct 2015.

[4] W. Shang, Q. Ding, A. Marianantoni, J. Burke and L. Zhang, "Securing building management systems using named data networking," *IEEE Network,* vol.28, no.3, pp.50-56, May-June 2014.

[5] W. Granzer, F. Praus and W. Kastner, "Security in Building Automation Systems," *IEEE Transactions on Industrial Electronics,* vol.57, no.11, pp.3622-3630, Nov. 2010.

[6] W. Granzer, W. Kastner, G. Neugschwandtner and F. Praus, "Security in networked building automation systems," *In Proc. 2006 IEEE International Workshop on Factory Comm. Systems,* pp.283-292, Torino, Italy.

[7] Q. Gou, L. Yan, Y. Liu and Y. Li, "Construction and Strategies in IoT Security System," *In Proc. IEEE Intl Conference on Green Computing and Communications*, pp.1129-1132, 20-23 Aug. 2013, Beijing, China.

[8] Z. Zhang, M. Cho, C. Wang, C. Hsu, C. Chen and S. Shieh, "IoT Security: Ongoing Challenges and Research Opportunities," *In Proc. the 7th IEEE International Conference on Service-Oriented Computing and Applications (SOCA),* , pp.230-234, 17-19 Nov. 2014, Matsue, Japan.

[9] F. Praus and W. Kastner, "Secure control applications in building automation using domain knowledge," *In Proc. the 8th IEEE International Conference on Industrial Informatics (INDIN),* pp.52-57, 13-16 July 2010, Osaka, Japan.

[10] X. Xingmei, Z. Jing and W. He, "Research on the basic characteristics, the key technologies, the network architecture and security problems of the Internet of things," *In Proc. the 3rd Int. Conf. on Computer Science and Network Technology (ICCSNT)*, 12-13 Oct. 2013, Dalian, China.

[11] S. Chen, H. Xu, D. Liu, B. Hu and H. Wang, "A Vision of IoT: Applications, Challenges, and Opportunities with China Perspective," *IEEE Trans. on Internet of Things*, vol.1, no.4, pp.349-359, Aug. 2014.

[12] W. Khamphanchai., A. Saha, K. Rathinavel, M. Kuzlu, M. Pipattanasomporn, S. Rahman, B. Akyol and J. Haack, "Conceptual architecture of building energy management open source software (BEMOSSTM)," *In Proc. the IEEE PES Innovative Smart Grid Technologies Conference (ISGT-Europe),* 12-15 Oct. 2014, Istanbul, Turkey.

[13] W. Khamphanchai, M. Pipattanasomporn, M. Kuzlu and S. Rahman, "An agent-based open source platform for building energy management," *In Proc. the IEEE PES Innovative Smart Grid Technologies Conference (ISGT-Asia)*, 3-6 Nov.2015, Bangkok, Thailand.

[14] J. Haack, B. Akyol, B. Carpenter, C. Tews, and L. Foglesong, "VOLTTRON: An agent platform for smart grid." *In Proc. the 4th International Workshop on Agent Technologies for Energy Systems*, pp.1367-1368, May 10, 2013, Minnesota, USA.

[15] CIA Triad. [Online]. Available: http://www.cyber-51.com/cyber51-blog/145-cia-triad. Retrieved: Oct 2015.

[16] OWASP Top Ten 2013 [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#tab=OWASP_Top_10_for_2013. Retrieved: May 2015.

[17] The Galois / Counter Mode of Operation (GCM) [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf. Retrieved: Oct 2015.

[18] M. Matsumoto and T. Nishimura, "Mersenne twister", *ACM Transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3-30.

[19] PBKDF2 Algorithm [Online]. Available: https://tools.ietf.org/html/rfc2898#appendix-A.2. Retrieved: Oct 2015

[20] Systemd "System and Service Manager" [Online]. Available: http://www.freedesktop.org/wiki/Software/systemd. Retrieved: May 2015.