WS3H: Webified Self-Sovereign Smart Home

Valentin Siegert[0000−0001−5763−8265], Clemens Albrecht[0000−0002−2650−0447], Mahda Noura[0000−0002−5105−2463], and Martin Gaedke[0000−0002−6729−2912]

Distributed and Self-organizing Systems,
Chemnitz University of Technology, Chemnitz Germany
{valentin.siegert,mahda.noura,martin.gaedke}@informatik.tu-chemnitz.de
clemens.albrecht@s2017.tu-chemnitz.de

Abstract. The Web of Things (WoT) concept proposes to integrate the existing Web ecosystem with Things to provide an interoperable infrastructure which goes beyond basic network connectivity. Existing approaches unify the process to integrate the devices into the Web by automatically generating Web APIs based on semantic device descriptions. However, they are either limited to specific protocols or do not address transport security and authorization. In this paper, we demonstrate our approach on a Webified Self-Sovereign Smart Home that uses self-sovereign identity to establish transport security and authorization in a webified smart home. Our solution provides end user support with easy deployment and addresses the local scope of a smart home due to its self-sovereignty.

Keywords: Self-Sovereign Identity · Smart Home · Web of Things

1 Introduction and Related Work

Homes are becoming *smarter* by integrating Internet of Things (IoT) devices to improve the user comfort through automation. Integrating IoT devices to the Web – called the Web of Things (WoT) – provides an interoperable infrastructure which simplifies their accessibility allowing to build IoT applications on top of Web API's [5]. Although the WoT working group[1] advocates exposing physical objects to the Web, they do not deal with the actual development of the Web interface, which is a significant barrier to fast prototyping of WoT applications. On the other hand, solutions like Matter[2] and WoTDL2API [5] propose an abstraction layer to automate the process of integrating IoT devices into the Web. Matter is an industry standard that builds on established technologies, all of which use Internet Protocol version 6 (IPv6). WoTDL2API on the other hand provides interoperable access to IoT devices by using a model-driven process to generate automatically RESTful APIs from WoTDL instances.

Securing smart homes is in the end user's best interest, even though they may not think about security until after they purchase smart devices [3]. Unfortunately, the Mirai Botnet is only one example that insecure IoT devices can

[1] https://www.w3.org/WoT/wg/
[2] https://buildwithmatter.com/

cause huge damage outside their owners interest [1]. Matter establishes unified APIs with a standardized security framework, but only supports devices of IPv6. Despite the motivation for a larger address range with IPv6, the rollout is not yet complete according to Google statistics[3]. WoTDL2API [5] is an application layer protocol that utilizes HTTP and can therefore also work with private IPv4 addresses. It is thus more suitable for Smart Homes, however it lacks support for transport security and authorization.

We therefore propose the **Webified Self-Sovereign Smart Home (WS3H)**; a secure, private and easy to use WoT solution for the Smart Home. It utilizes WoTDL2API [5] and Self-Sovereign Identity (SSI) for authentication and authorization based on the proposal for SSI in IoT [4]. Besides SSI, the two most prominent approaches for transport security and authorization are TLS [6] and Wireguard [2]. However, both impose a burden on end users because their initial adoption is difficult, which is easier in WS3H. TLS requires the establishment of a private CA or the use of a public CA, and in Wireguard the public key exchange of all smart home devices must be implemented by the end user. Our solution surpasses TLS and Wireguard by its ease of use for end users who do not want to deal with certificates or public and private keys.

The remainder of the paper presents WS3H and describes a demonstration on how a user can make use of WS3H within his home by connection a new device to WS3H.

2 Webified Self-Sovereign Smart Home (WS3H)

WS3H provides a secure and private solution. The application design of WS3H follows a hub-and-spoke topology and realizes SSI on behalf of standards provided by Hyperledger Aries[4]. The hub device provides the blockchain used by the SSI protocols. Furthermore, both the hub and the spoke devices provide an implementation of a Hyperledger Aries agent. The authentication and authorization of a new IoT device follows a three step procedure.

1. A spoke device starts a RFC 0160 Connection Protocol flow. It sends the hub its per-connection generated DID and public key, and the hub responds with its corresponding ones. From this point on, both peers can communicate over a DIDcomm-secured channel.
2. Then, the spoke device initiates RFC 0036 Issue Credential. After the hub receives the connection request, the only user interaction is to use notice and consent on whether to allow the spoke device to receive the credential. Using the device description provided by the spoke device, he can identify if it is the device he just connected to the network. Upon approval, the speaking device receives the credential, which it can now use to authenticate itself.
3. Using the Verifiable Credential (VC) it received in the previous step, the spoke device now starts the RFC 0037 Present Proof Protocol. It sends a

[3] `https://www.google.com/intl/en/ipv6/statistics.html`
[4] `https://github.com/hyperledger/aries-rfcs`

verification request to the hub, which responds with a request for a matching credential. When the credential is verified, the hub confirms the confirmation and thus the spoke device is considered authenticated.

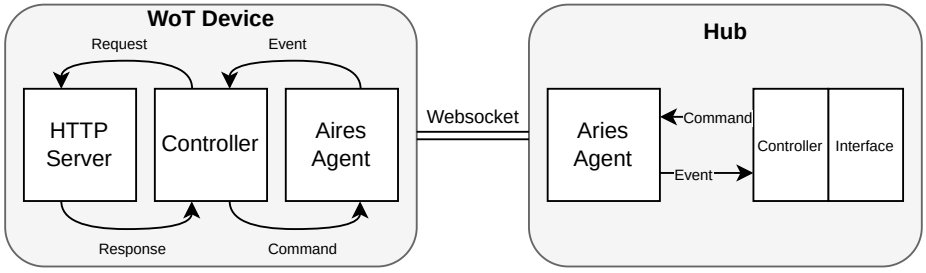

Fig. 1. WS3H Architecture

Figure 1 shows the architecture developed for WS3H. It shows the IoT device consisting of the WoTDL2API server, as well as the Aries Controller and Agent. The hub contains an agent and the Web server that performs the role of a controller and is responsible for presenting the Web UI.

3 Demonstration

The WS3H demo shows a typical smart home deployment scenario with two hosts. A Raspberry Pi connected to a light sensor, a motion sensor, and a volume sensor, and a laptop with a browser displaying the hub's UI. It demonstrates the use case of a dashboard that presents data about the user's home. The Raspberry Pi and the Laptop feature a full implementation of the hub-and-spoke concept with one spoke device. Both devices contain an implementation of an agent and a controller, and the Raspberry Pi contains the WoTDL2API server. To realize the concept, the following four components were used.

- The hub and the spoke device contain an Aries Cloudagent Python. This is a Python implementation of a full Hyperledger Aries Agent. It was chosen because it is currently the most complete implementation of such an agent. This agent provides the implementation of the protocols used to communicate between the two devices.
- To provide the blockchain implementation, the hub contains an implementation of the Hyperledger Indy blockchain using a VON Network Node Docker image. The Hyperledger Indy blockchain is currently the only supported option in Hyperledger Aries.
- The communication between the agent and the WoTDL2API server on the spoke device is handled using a custom Python application using Flask. It handles the packing and unpacking of the HTTP request and response, as well as performing the task of an agent controller.

– A custom Python application using Django implements the UI presented to the user. The UI implements the management of connected spoke devices as well as presenting the retrieved data to the user. This application also provides the agent controller implementation on the hub device as well as the packing and unpacking of the HTTP request and response.

Using this demo, the user can process connecting the Raspberry Pi to the network, which appears in the hub's UI. The user can then allow the connection of the device and the issuing of the credential. If these steps were completed successfully, the user can securly inspect the data provided by the sensors. A video[5] of the user's perspective on the dashboard UI presents how easy our solution is for the user after connecting a device to his Smart Home.

4 Conclusion

We demonstrated in this paper our Webified Self-Sovereign Smart Home (WS3H) approach, which secures Smart Homes' transport layer and establishes device authorization. It extends WoTDL2API [5] with a self-sovereign identity approach, targeting the use case of Smart Homes and its local scope. To establish secure channels within WS3H, the concept of self-sovereign identity for IoT devices [4] is implemented by using secure DIDcomm via VCs with the usage of Hyperledger Aries and Indy. The demonstration includes an example run of connecting a new device in a Smart Home and issuing credentials for it with ease for end users.

References

1. Antonakakis, M., April, T., Bailey, M., et al.: Understanding the mirai botnet. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 1093–1110. USENIX Association, Vancouver, BC (Aug 2017), https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis
2. Donenfeld, J.A.: Wireguard: next generation kernel network tunnel. In: NDSS. pp. 1–12 (2017). https://doi.org/10.14722/ndss.2017.23160
3. Emami-Naeini, P., Dixon, H., Agarwal, Y., et al.: Exploring how privacy and security factor into iot device purchase behavior. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. p. 1–12. CHI '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3290605.3300764
4. Fedrecheski, G., Rabaey, J.M., Costa, L.C.P., et al.: Self-Sovereign Identity for IoT environments: A Perspective. In: 2020 Global Internet of Things Summit (GIoTS). pp. 1–6. IEEE, Dublin, Ireland (Jun 2020). https://doi.org/10.1109/GIOTS49054.2020.9119664
5. Noura, M., Heil, S., Gaedke, M.: Webifying Heterogenous Internet of Things Devices. In: Web Engineering, vol. 11496, pp. 509–513. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-19274-7_36
6. Rescorla, E.: The transport layer security (tls) protocol version 1.3. Tech. rep. (2018), https://www.rfc-editor.org/rfc/rfc8446

[5] https://youtu.be/-jrfGlrHfAc