A Universal Controller to Take Over a Z-Wave Network

Loïc Rouch, Jérôme François, Frédéric Beck, Abdelkader Lahmadi

A Universal Controller to Take Over a Z-Wave Network

Loïc Rouch, Jérôme François, Frédéric Beck
Inria Nancy - Grand Est, Villers-lès-Nancy, France

Abdelkader Lahmadi
LORIA - University of Lorraine, France

Abstract

Home automation systems adoption rapidly increases with the growth of Internet of Things (IoT). IoT devices are often equipped with wireless communication capabilities including WiFi, Z-Wave, or Zigbee to be remotely controlled and fit any home. They become thus natural targets for potential cyber-attacks, with the intent to take control over them and to eventually expose end-users to privacy, security and safety risks. However, realizing such attacks usually requires expert knowledge and costly hardware including Software-Defined Radio platforms for packets sniffing, spoofing, and injection.

In this paper, we demonstrate that off-the-shelf hardware is sufficient to take over any Z-Wave network without knowing its topology or compromising any original devices and remaining unnoticeable for the primary controller. Our attack consists in building an adversary Z-Wave universal controller by reprogramming a mainstream USB stick controller. The technique exploits two features provided by the USB stick which allow (1) to set the network identifier (HomeID) and (2) learn many devices identifiers even if they are not physically available.

1 Introduction

According to Gartner [9], there will be more than 20 billion Internet of Things devices in 2020, which represents a $ 3,010 B market. Their deployment and integration in existing systems becomes easier, requiring (existing) wires only for power, or even no wire when using battery powered devices. Many of these devices have adopted communication standards like ZigBee, Bluetooth Low Energy (BLE), WiFi, or Z-Wave to make them remotely controllable and easy to use. One of the primary application domains of these devices is home automation. To satisfy end-users convenience, they are usually plug-and-play and therefore easy to use and deploy. The user only needs to plug-in, switch on and do almost zero configuration, usually by simply pressing a single button for pairing the devices with a primary controller. The IoT market being large and competitive, manufacturers are driven by acquiring market shares as rapidly as possible. As a consequence, development cycles are shortened with a focus on end-user functionalities. Several works highlight successful attacks on IoT devices, partly due to design compromises, sometimes leading to weak security while improving usability and reducing prices.

This introduces an opportunity for attackers to exploit vulnerabilities on either the embedded software or hardware, or the communication protocol itself. However, building offensive tools at the wireless communication level remains a challenging task because it requires costly hardware, a strong knowledge about the device, its protocol and its capabilities. It may even need to apply reverse engineering techniques on the embedded firmware. The severity of the attack depends upon what control commands the attacker is able to access and how many devices he can take control over. One of the most severe attacks is to take control over all of the deployed networks in a home automation system while remaining hidden, which allows the attacker to access sensitive information (*e.g.* reading motion detectors), to cause denial of services (*e.g.* deactivating alarm systems), or to build further attacks (*e.g.* temperature manipulation to create a covert communication channel [6]).

In this work, we focus on Z-Wave because of its popularity [4] and characteristics. It allows manufacturers to build products with reasonable prices and advanced functionalities. However, when building products based on the official Z-Wave stack,

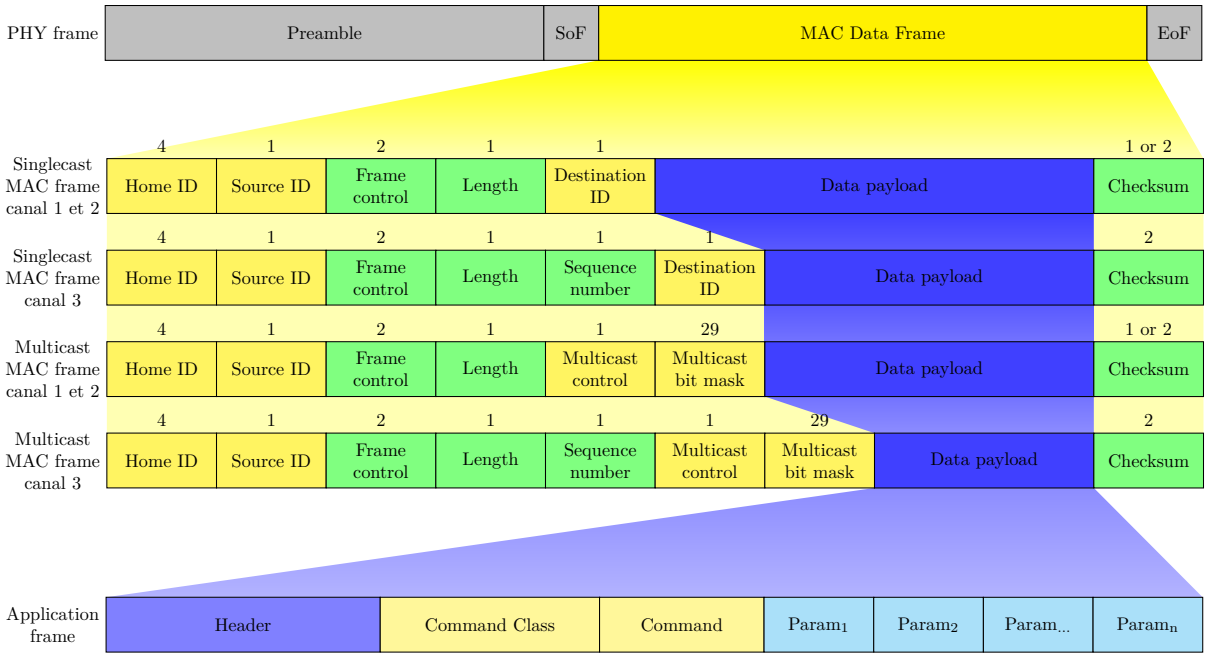

Figure 1: Format of a Z-Wave frame

manufacturers may combine functionalities offered by the protocol itself with their own features. On one hand, the Z-Wave protocol leverages automatic network discovery. On the other hand, to ease the replacement of a controller, several manufacturers provide a backup and restore functionality. Although passively listening a Z-Wave network to extract its network identifier with mainstream device is already known, this paper goes further by building a universal controller from a mainstream Z-Wave controller. As a result, it is possible to control all devices of a Z-Wave network even those not observed during the passive listening phase. Hence, taking over any Z-Wave network is practicable at low cost.

The remainder of this paper is organized as follows. Section 2 describes the Z-Wave protocol. Our approach and method for building the adversary controller are detailed in Section 3. Additional remarks are discussed in Section 4, while an overview of related work is given in Section 5. Finally, Section 6 concludes the paper.

2 Z-Wave overview

Z-Wave is a wireless protocol developed by the company ZenSys in 1999, and acquired by Sigma Designs in 2008. The current (5th) generation of Z-Wave chips, also called 500 Series, has been on the market since 2013. The Z-Wave protocol is based on the ITU-T G.9959 standard [7], but remains a proprietary protocol. Only recently, a documentation of the protocol has been made publicly available by Sigma Designs [16]. Even if the protocol addresses security concerns, security by obscurity was one of the used archetypes, as signing Sigma Designs' NDA forbids any reverse engineering initiatives or studies on the protocol. A notable initiative to provide an open source implementation of Z-Wave is OpenZWave [12] which aims at making the protocol specification freely available. OpenZWave provides a library, available in multiple programming languages (including C and Python), and an API to control Z-Wave devices.

2.1 Identifiers

A Z-Wave network is built around two types of devices: controllers and slaves. The controller is the central entity of the network, and every devices in the network are linked to it. Multiple controllers can be part of a single installation if the area to cover is too wide, or if there are a large number of devices. A Z-Wave network relies on two types of identifiers: the HomeID, and NodeIDs.

A controller is identified by a unique value named the HomeID. Each controller has its own HomeID and this value is supposed to be unique, being set by the manufacturer when the controller is being built. A reset of the controller either keeps the value or changes it randomly. The HomeID has a length of

32 bits, which represents more than 4 B possibilities ($2^{32} = 4\,294\,967\,296$). Each device in a Z-Wave network uses this HomeID in each message sent, which makes it identifiable by any other device associated to the same network. Therefore, the HomeID of the controller serves as a unique network identifier.

Within a Z-Wave network, each device has a unique identifier called the NodeID. The controller always has the NodeID 1. The controller of a network assigns, during a pairing sequence, a NodeID to each device. Even if the NodeID has a length of 8 bits, there can only be 232 nodes (slaves) in a network, some NodeIDs being reserved.

2.2 Frame format

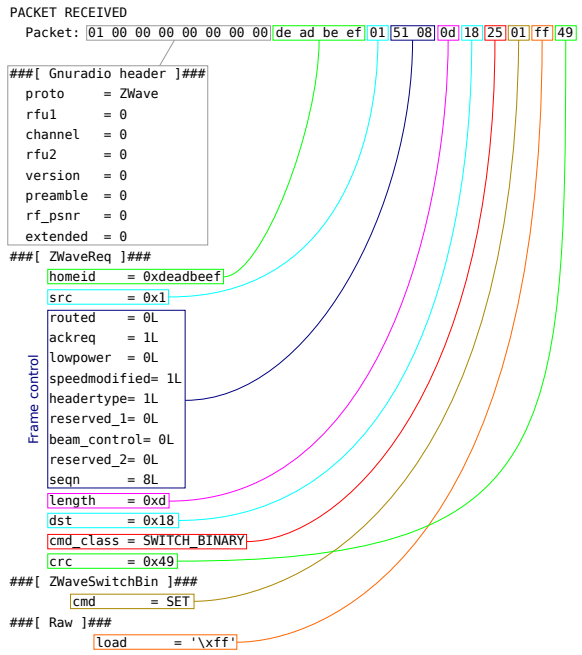

Figure 2: Z-Wave frame example, SWITCH_ON command

Figure 1 describes the format of a Z-Wave frame without use of encryption. At the physical layer, the preamble is used for synchronization while the MAC frame itself is delimited between SoF (Start-of-Frame) and EoF (End-of-Frame). Several formats are possible for the MAC Data Frame (unicast denoted as singlecast in Z-Wave, or broadcast denoted as multicast in Z-Wave) but all of them begin with the HomeID of the network, immediately followed by the NodeID of the device sending the message (Source ID). When the message is sent toward a specified node (singlecast), the NodeID of the receiver (Destination ID) is also specified.

Then, the data payload contains the Z-Wave command representing actions to be performed on the devices or returned values by the devices. Commands are grouped into command classes and the reader can refer to the specification[1] for details since this is independent of the attack described in this paper.

Figure 2 shows an exemple of a Z-Wave frame captured by GNURadio[2] and displayed by Scapy-radio[3]. As previously stated, the HomeID (`deadbeef`) of the network and NodeID of the emitting device (`0x01`: the controller) are present in the frame immediately after some preamble and header. After *frame control* and *length* fields, the NodeID of the receiving device is specified (`0x18`: NodeID 24). The command (`0x25`: SWITCH_BINARY) and value (`0xff`: ON), just before the ending checksum, requests the receiver, a smart plug, to be switched on.

2.3 Pairing procedure

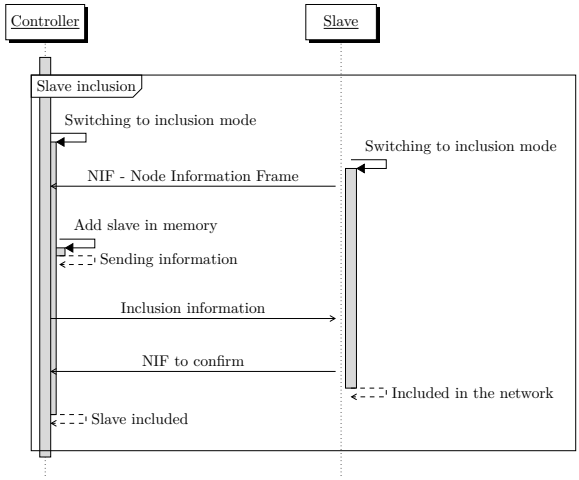

Figure 3: Inclusion of a device in a Z-Wave network

Association/Pairing is the main feature allowing to create a Z-Wave network. The controller keeps track of the paired nodes. There is a firmware limitation, required by the specification, allowing messages to be only sent to registered slaves/nodes. Indeed, the controller can not send a message to a device until it is paired with it and associated to its HomeID.

[1] Z-Wave specification: `http://z-wave.sigmadesigns.com/design-z-wave/z-wave-public-specification/`

[2] GNURadio: `https://www.gnuradio.org/`

[3] scapy-radio: `https://bitbucket.org/cybertools/scapy-radio/src`

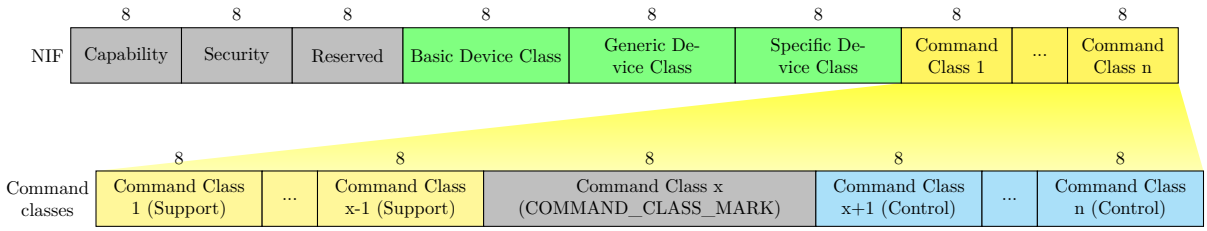

Figure 4: Format of a Node Information Frame (NIF)

In order to be part of a Z-Wave network, a slave device needs to go through a pairing/inclusion procedure. The controller and the slave node need to be placed into inclusion mode, which requires a physical action like pressing a button on the device. Once in inclusion mode, the controller listens for a Node Information Frame (NIF) emitted by the slave device, mainly indicating its capabilities (supported commands). The controller replies with the new NodeID assigned to this slave device and the HomeID of the network. To confirm that it is now part of the network, the slave sends another NIF. The overall procedure is summarized in Figure 3. The inclusion procedure is the primary phase to build a Z-Wave network, ensuring every device in the network uses the same HomeID and can communicate.

The NIF purpose is to ease configuration of Z-Wave devices, by providing a description of their capabilities. Figure 4 illustrates its format. It begins with protocol specific parts, followed by device capabilities. They are divided into multiple levels. The Basic Device Class defines the network role of the device in a Z-Wave network: a portable controller, a static controller, a slave, or a slave with routing capabilities. The Generic Device Class defines the real functionality of this device (*e.g.* alarm sensor, entry control). Some Generic device classes can be complemented by a Specific Device Class if needed (*e.g.* an alarm sensor can be a smoke detector, or an intrusion detector). Basic, Generic, and Specific device classes imply some predetermined Command Classes that are available for the device. After the device classes, the NIF contains Command Classes supported by the device, followed by a separator *(COMMAND_CLASS_MARK)*. If the device is able to directly control other devices (*e.g.* a light switch which directly controls a led bulb), the separator is followed by the list of Command Classes this device is able to control.

2.4 Auto-discovery

Auto-discovery makes every node in a network aware of its neighbors. Upon startup, the controller tries to reach every registered slave node to discover how the network is organized. Because Z-Wave networks are meshed, some nodes can relay messages to other nodes. This enables a wider coverage area around a single controller. When a slave node cannot be directly reached from the controller, another slave node relays the message, enabling the controller to communicate with it. If a device is moved to another place, this feature allows it to keep on communicating with the controller through intermediate devices.

When using auto-discovery, the controller tries to reach every known nodes, which will reply with a Node Information Frame (NIF) similarly to the process during inclusion mode triggered by a physical action.

This makes a Z-Wave network extremely robust against topology modifications. Starting with a simple list of registered nodes, a controller is able to discover the capabilities of all nodes, and how nodes are physically organized.

2.5 Z-Wave security

Z-Wave devices are often closely related to the user activity by being part of home automation systems. Therefore, they gather personal information, and usually control sensitive appliances. The security of these devices is even more important when they are part of physical security functions, like access control or alarms. Z-Wave security takes place at different levels.

First, the HomeID uniquely identifies a network and the pairing process to include new nodes in the network requires an explicit action on the network's controller. However, the communication occurs in clear by default, which allows anyone within the radio range of the network to observe and decode the messages.

Second, Sigma Designs introduced the 5th generation of Z-Wave in 2013. The 500 Series remains

backward compatible with previous generations, but introduces new features and improves security. In the newer Z-Wave generations (400 and 500 series), manufacturers can implement a secure mode by using hardware-based 128-bit AES encryption capabilities. When devices are paired in secure mode, a key is exchanged between the node and the controller, and it is used to encrypt messages.

Although secure mode and use of encryption could solve many security issues, its support has been recently added leading to limitations of its proper adoption. Indeed, buying a 5th generation, Z-Wave Plus (Z-Wave+) certified product, does not guarantee the implementation of the secure mode which may lead to confusion for users. Assuming a device manufacturer implements it, a certified Z-Wave+ device has to be backward compatible with former Z-Wave devices, *i.e.* without security. The end-user is thus responsible to choose whether or not he wants to use the secure mode. Since users are primarily looking for functionality and using devices as plug-n-play, documentation is overseen and co-existence of two modes may lead to misunderstandings.

3 Attack procedure

The objective of our attack is to take the full control of a Z-Wave network without the use of costly tools. Indeed, our goal is to create a universal controller from an off-the-shelf controller.

Our attack scheme is summarized as follows:

- Phase 1: Associate the adversary controller with all possible nodes
- Phase 2: Capture the HomeID of the targeted network to be used by the adversary controller
- Phase 3: Copy the captured HomeID to the adversary controller
- Phase 4: Restart the adversary controller

However, by default, it is impossible to:

- Associate a node without physical user action
- Set the HomeID of a controller to a specific value

To overcome these difficulties, the attack consists in the following steps: (1) use a single adversary-controlled device to pre-fill the universal controller in phase 1; (2) use a state-of-art technique in phase 2 relying on SDR (Software-Defined Radio) device [14] or even on a inexpensive DVB-T tuner[4]; (3) leverage backup/restore feature of the adversary controller for realizing phase 3.

3.1 Abused features

Z-Wave features described in Section 2 enable the protocol to be robust to different or changing network topologies. Auto-discovery and Node Information Frame (NIF) give the network introspection and adaptability capabilities. However, in case of a controller failure, re-building the network can be tedious because of the manual pairing process requiring physical actions from the users. Moreover, devices can be numerous and hardly accessible. To circumvent these issues, some manufacturers implemented a backup/restore procedure[5]. A user should backup its controller to easily replace it in case of failure without the necessity to redo the pairing with all slave nodes. Once a controller has been backed-up, restoring it on a new controller makes the new one identical to the former one and thus directly usable by the user.

In addition, the auto-discovery and NIF could also be used by an adversary-controller to discover device capabilities without any knowledge about it, but still requiring physical access to them for pairing purposes.

In our experimental tests, by using a full backup/restore procedure while modifying the HomeID in the backup, only the HomeID could be backed-up from our mainstream controller excluding thus the full network state (NodeIDs and capabilities).

Backups could be stolen to be restored on a non-legitimate controller, but this supposes unsafe storage of backups or physical access to them. Hence, our attack does not suppose neither prior access to any backup nor any physical access to target Z-Wave devices. Only a single mainstream adversary controller and a single adversary slave are needed.

3.2 Exploit details

Figure 5 illustrates the different steps of the attack except Phase 1. Indeed, Phase 1 (Universal Controller Creation) has to be done only once but will be described in Section 3.2.2. In a nutshell, this

[4]Lot of resources on: `http://www.rtl-sdr.com`, more on turning a DVB-T tuner in a SDR on: `http://www.epanorama.net/newepa/2013/10/23/software-defined-radio-with-usb-dvb-t-stick/comment-page-2/`.

[5]Not all controllers implement this feature. Due to ethical reasons, the model of used controller cannot be revealed.

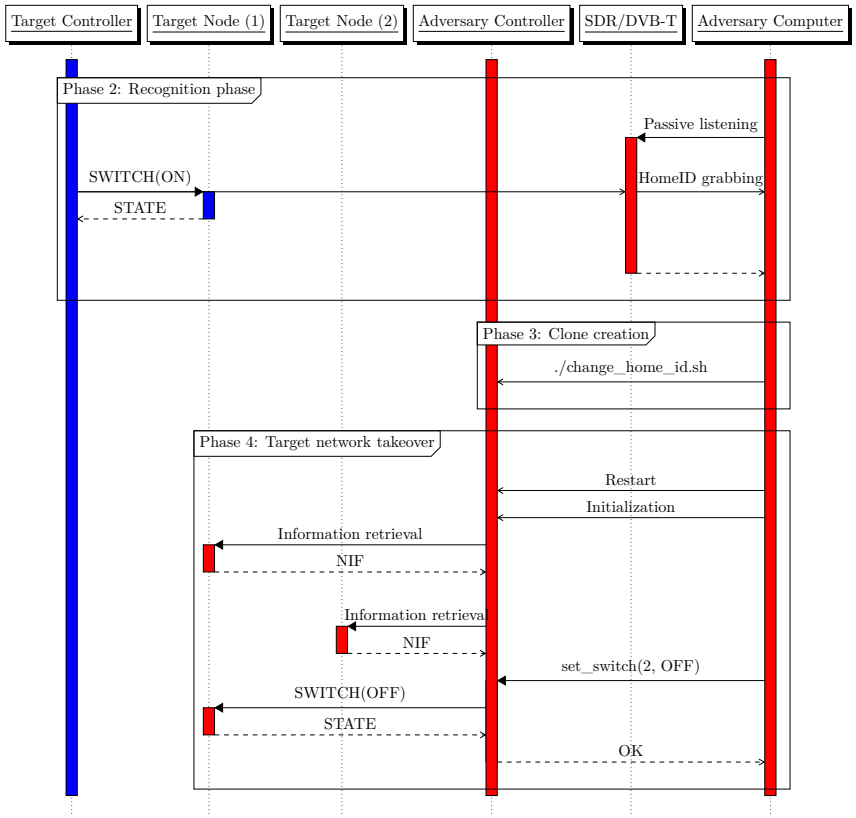

Figure 5: Attack phases

consists in forcing and repeating pairing as shown in Figure 3.

3.2.1 Capturing and copying the HomeID

The HomeID being an important property of a Z-Wave network, to gain access to a target network, we must discover the used identifier (Phase 2 in Figure 5). It is shared by all devices, used in all messages, and it is uniquely set in controllers. We used a DVB-T receiver (USB SDR based on RTL2832U and R820T2 chips) in conjunction with Waving-Z[6], listening for any message in the target network. Once received, we can extract the HomeID of the target network.

Once the target network HomeID captured, the next step is to assign it to the adversary controller. This operation is strictly forbidden by the protocol, to prevent users to choose a predictable HomeID, or use the same one as their neighbors. As an example, the OpenZWave library we use to send commands to our USB connected controller does not provide any instruction to arbitrarily change the HomeID of our controller.

[6]Waving-Z: `https://github.com/baol/waving-z`

When doing a hard reset on the controller, the HomeID changes to a new randomly chosen value, which is an impractical solution to associate a user-defined HomeID on the adversary controller.

We thus use the restore feature of the adversary controller to clone the originally captured HomeID. Figure 6 shows an extract of the restoration logs originally used to create our script mentioned in Phase 3 in Figure 5. There are two *SENDING* lines containing the two commands sent to the controller to respectfully change the HomeID and apply/save the modification. In this example, the HomeID being set is `0xDEADBEEF` which can be seen at the end of the first command, right before the checksum `F6`. Sending those two commands have the effect of setting the wanted HomeID without ordering a reset of the controller, hence keeping already registered nodes.

3.2.2 Universal controller creation

To be able to send commands to the nodes of the target network, they need to be known by our controller. As mentioned before, there is a firmware limitation preventing the controller to send messages to

```
[2017-11-22 17:55:42.926] [D] [zway] SENDING: ( 01 0C 00 2B 00 00 08 00 04 DE AD BE EF F6 )
[2017-11-22 17:55:42.927] [D] [zway] RECEIVED ACK
[2017-11-22 17:55:42.936] [D] [zway] RECEIVED: ( 01 04 01 2B 01 D0 )
[2017-11-22 17:55:42.936] [D] [zway] SENT ACK
[2017-11-22 17:55:42.936] [I] [zway] Job 0x2b (Write bytes to extended EEPROM): Done
[2017-11-22 17:55:42.936] [D] [zway] Job 0x2b (Write bytes to extended EEPROM): success
[2017-11-22 17:55:42.956] [I] [zway] Removing job: Write bytes to extended EEPROM
[2017-11-22 17:55:42.956] [D] [zway] SENDING: ( 01 25 00 2B 00 05 80 00 1D 01 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 68 )
[2017-11-22 17:55:42.959] [D] [zway] RECEIVED ACK
[2017-11-22 17:55:42.966] [D] [zway] RECEIVED: ( 01 04 01 2B 01 D0 )
[2017-11-22 17:55:42.966] [D] [zway] SENT ACK
[2017-11-22 17:55:42.966] [I] [zway] Job 0x2b (Write bytes to extended EEPROM): Done
[2017-11-22 17:55:42.966] [D] [zway] Job 0x2b (Write bytes to extended EEPROM): success
[2017-11-22 17:55:42.986] [I] [zway] Removing job: Write bytes to extended EEPROM
```

Figure 6: HomeID restoration

a NodeID which is not paired with the controller.

Similarly to what is described in Section 3.2.1, we could continue listening for other messages to discover which slave nodes are present. Unlike regular networks, low-powered devices limits the use of keep-alive mechanisms, in particular when devices are battery-powered. Z-Wave standard does not require such a mechanism by default. As a result, waiting for messages of all devices can be long (several hours or days). A unobserved NodeID does not imply nonexistence of the node because the latter may not have emitted any message yet. Therefore, our method creates a universal controller, *i.e.* with all 232 possible NodeID pre-filled, as highlighted in the preamble of Section 3.2.

We initially expected to create our own backup file with all NodeIDs and rely on the restore feature to configure the controller. However, all previously paired nodes have been removed after restoring our backup. Indeed, it only allows to create a blank controller with the same HomeID as our target network. We thus use a single adversary slave to be registered as multiple devices (from 2 to 232, 1 being the controller). Normal pairing procedure is thus performed but requires resetting the slave node in order to make it appear as a new device (otherwise, the former NodeID is considered as expired and removed from NodeID list of the controller).

This method allowed us to populate the whole list of paired nodes in our adversary controller. Changing the HomeID with a full backup/restore procedure would result in the loss of every registered node. As including them is time consuming and quite tedious, pairing again 232 times the same device is not an option. However, we use partial restore functionality introduced in Section 3.2.1, which only changes the HomeID.

As a result, our adversary controller is universal, as it can take over any Z-Wave network by only requiring to change the HomeID. Hence, this step (universal controller creation) is done only once and independent of the HomeID.

3.2.3 Taking over the target Z-Wave network

Once the adversary controller configured with the cloned HomeID and the full list of NodeID, a simple restart will allow the attacker to control the network.

Indeed, our attack leverages automatically the powerfulness of auto-discovery and NIF features that are legitimately used at startup by our adversary controller.

Therefore, the controller starts by probing registered devices (in our case, all possible ones). The real ones then reply with a NIF frame as shown in Phase 3 in Figure 5, making our controller aware of their existence and usable commands. Otherwise, a timeout will expire and non-answering nodes will be marked as unavailable. However, as those nodes are kept in the controller memory, if a new node is added to the target network, we will immediately be able to control it. Our adversary controller will mark the new node as alive when observing the *NIF to confirm* message described in Figure 3.

For each inexistent node, the controller waits until a timeout of 3 seconds is reached. Then, the device is probed again with the same timeout. The maximum theoretical startup time therefore is $232 * 3 * 2 = 1392s = 23.2m$. After the startup process, we have a fully functional clone of the target network controller.

4 Discussion

In summary, the required steps to customize our universal controller to gain access to a target network are: passive listening to discover the target network HomeID, set it in our controller, restart our controller, and let it scan the target network. We can use a DVB-T tuner to do the listening part and an off-the-shelf controller for the adversary controller which highly reduces the hardware cost and the technical skills to perform the attack, making the overall cost of building such controller below $ 100.

Except scanning the full potential node identifiers, our universal controller acts as a legitimate one. Hence, once it is in the network, it is impossible to detect it based on the types and formats of commands sent. Also, the use of multiple controllers is allowed with Z-Wave to overcome the limitation of the number of slaves associated to a controller. Therefore, advanced techniques could be used such as behavioral approaches, eventually with machine learning, to track unexpected sequences of actions for example (similar to anomaly detection in IP networks) but those will be obviously only reserved for expert users which is contradictory with the wider range of targeted IoT users population.

A natural counter-measure against our attack procedure is to enable the secure mode with encryption to prevent our universal controller from taking over a node paired in secure mode. However, as demonstrated in [3], manufacturers implementations of the secure mode may introduce security flaws. A new version of the secure mode was recently published. It may solve previous issues but won't apply to the majority of existing installations. Moreover, as highlighted in Section 2.5, backward compatibility concerns and priority to usability slow down the adoption of secure mode.

5 Related work

Many studies have investigated the security of IoT devices and networks to demonstrate their remote compromission, providing attackers with the ability to remotely abuse them to make DDoS attacks, or to disable and modify their operations [2]. To assess the risks, the authors in [13] designed an IoT honeypot to discover Telnet-based attacks targeting various IoT devices running on different CPU architectures. In [18], the authors studied Sybil attacks in IoT. Although such an attack occurs at the application or data-level, ours can be considered as a network-level sybil attack because an adversary controller spoofs the network identity and makes actions on behalf of the primary controller. In [3], authors have identified a security flaw in a door lock, allowing them to remotely reset its network key, used to do encryption, to a value of their choice. The used vulnerability is not in the Z-Wave protocol, but it is related to the manufacturer implementation.

In [11], the authors demonstrated an attack on compact fluorescent lamps (CFL) which are controlled by a Z-Wave enabled light dimmer. They show that such attacks could cause physical harm to home occupants through the explosion of the CFLs by switching it on and off alternatively. To make their attack, they exploited known vulnerabilities in two Z-Wave devices. Our exploit method eases the attack by alleviating the need of compromising the controller or the device, since our adversary controller allows him to take control of the network and make any required action for such attacks.

Although, previous references are selected examples of security threats in IoT devices, not exclusively limited to Z-Wave, solutions to improve security have been widely covered in particular in the area of wireless sensor networks mainly by focusing on open protocols such as 6LowPan [10]. Enabling security can be derived at the different layers, from physical to application, as highlighted in [5]. Primary concerns are devoted to enable cryptographic mechanisms for confidentiality, authentication, integrity or non-repudiation, especially assuming low resources [8]. Related to this topic is the key management protocols [15]. Attacks against the proper functioning of the network itself have been widely studied, especially in the context of open environments, with attacks targeting routing and their counter-measures [1].

Although solutions have been proposed and proved secure several years ago, their real use in mainstream products is still fairly low. Such an observation has also been made by authors in [17] who advocates for network-level monitoring and mitigation to track suspect behaviors.

Even if most of these works are not focused on Z-Wave, the secure mode of Z-Wave leverages also cryptographic techniques. However, as shown in previous section, they are not widely adopted yet (backward compatibility and usability concerns) but Z-Wave benefits from a limited open specification and a restricted set of capabilities to be implemented in certified products.

Since Z-Wave is a wireless protocol, it is also subject to basic networking attacks including replay attacks or packet forging. However, advanced knowledge in signal analysis, processing and electronics can circumvent the latter. In Picod et al.'s [14], the

authors use a Software-Defined Radio (SDR) to listen, forge/create, and send Z-Wave packets. Their attack requires costly hardware (the SDR they used, a USRP B210, costs over $1,200) and advanced knowledge in signal processing. They made publicly available their used GNURadio schema for the processing of Z-Wave signals, but fine tuning it to work in any environment is complicated.

6 Conclusion

By leveraging two legitimate features: one from the Z-Wave specification (network discovery) and one device-specific (backup/restore), we demonstrated the feasibility to take over a Z-Wave network. The core part of our exploit consists in pre-building a universal controller using only two regular devices (a controller and a slave) which has then be only configured regarding the targeted network whose identifier has been passively captured and then restored. It is worth to mention that the attack can be performed without special hardware, with limited knowledge in radio and electronic, and with a limited budget (below $100). As IoT devices are more and more used and trusted in our daily life, their security will become more and more critical, especially when attacks are made up from a combination of legitimate features.

Indeed, the manufacturer of the controller we exploited has been notified but the issue will not be corrected since the backup-restore is a legitimate functionality itself.

We planned to further research on the protocol by looking more closely at the encryption part (secure mode), especially on a hardware level.

7 Acknowledgments

This work was partially funded by Region Lorraine under CPER Cyber-Entreprises framework. It is also supported by the High Security Lab hosted at Inria Nancy Grand Est.

References

[1] Airehrour, D., Gutierrez, J., and Ray, S. K. Secure routing for internet of things: A survey. *Journal of Network and Computer Applications 66* (2016), 198 – 213.

[2] Dhanjani, N. *Abusing the Internet of Things: Blackouts, Freakouts, and Stakeouts*. O'Reilly Media, 8 2015.

[3] Fouladi, B., and Ghanoun, S. Security evaluation of the Z-Wave wireless protocol. *Black Hat USA Conference 24* (2013).

[4] Google Trends. Google Trends - Z-Wave, ZigBee. `https://trends.google.com/trends/explore?q=%2Fm%2F07xy17,%2Fm%2F01brgt`.

[5] Granjal, J., Monteiro, E., and Silva, J. S. Security for the internet of things: A survey of existing protocols and open research issues. *Communications Surveys Tutorials 17*, 3 (2015), 1294–1312.

[6] Guri, M., Monitz, M., Mirski, Y., and Elovici, Y. Bitwhisper: Covert signaling channel between air-gapped computers using thermal manipulations. In *Proceedings of the 2015 IEEE 28th Computer Security Foundations Symposium* (Washington, DC, USA, 2015), CSF '15, IEEE Computer Society, pp. 276–289.

[7] ITU Telecommunication Standardization. ITU-T G.9959.

[8] Lee, J. Y., Lin, W. C., and Huang, Y. H. A lightweight authentication protocol for internet of things. In *International Symposium on Next-Generation Electronics (ISNE)* (May 2014), pp. 1–2.

[9] Meulen, R. v. d. Gartner Says 6.4 Billion Connected, Nov. 2015.

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

[11] Oluwafemi, T., Kohno, T., Gupta, S., and Patel, S. Experimental security analyses of non-networked compact fluorescent lamps: A case study of home automation security. In *Proceedings of the LASER 2013 (LASER 2013)* (Arlington, VA, 2013), USENIX, pp. 13–24.

[12] OpenZWave. OpenZWave website. `http://www.openzwave.com/`, Mar. 2017.

[13] Pa, Y. M. P., Suzuki, S., Yoshioka, K., Matsumoto, T., Kasama, T., and Rossow, C. Iotpot: Analysing the rise of iot compromises. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)* (Washington, D.C., 2015), USENIX Association.

[14] Picod, J., Lebrun, A., and Demay, J. Bringing software defined radio to the penetration testing community. In *Black Hat USA Conference* (2014).

[15] Roman, R., Alcaraz, C., Lopez, J., and Sklavos, N. Key management systems for sensor networks in the context of the internet of things. *Computers & Electrical Engineering 37*, 2 (2011), 147 – 159.

[16] Sigma Designs. Z-Wave Public Specification. `http://z-wave.sigmadesigns.com/design-z-wave/z-wave-public-specification/`, Aug. 2016.

[17] Sivaraman, V., Gharakheili, H. H., Vishwanath, A., Boreli, R., and Mehani, O. Network-level security and privacy control for smart-home iot devices. In *International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (2015), IEEE.

[18] Zhang, K., Liang, X., Lu, R., and Shen, X. Sybil attacks and their defenses in the internet of things. *IEEE Internet of Things Journal 1*, 5 (Oct 2014), 372–383.