Evaluating ITU-T G.9959 Based Wireless Systems Used in Critical Infrastructure Assets

Christopher Badenhop, Jonathan Fuller, Joseph Hall, Benjamin Ramsey, Mason Rice

Chapter 13

EVALUATING ITU-T G.9959 BASED WIRELESS SYSTEMS USED IN CRITICAL INFRASTRUCTURE ASSETS

Christopher Badenhop, Jonathan Fuller, Joseph Hall, Benjamin Ramsey and Mason Rice

Abstract ITU-T G.9959 wireless connectivity is increasingly incorporated in the critical infrastructure. However, evaluating the robustness and security of commercially-available products based on this standard is challenging due to the closed-source nature of the transceiver and application designs. Given that ITU-T G.9959 transceivers are being used in smart grids, building security systems and safety sensors, the development of reliable, open-source tools would enhance the ability to monitor and secure ITU-T G.9959 networks. This chapter discusses the ITU-T G.9959 wireless standard and research on ITU-T G.9959 network security. An open-source, software-defined radio implementation of an ITU-T G.9959 protocol sniffer is used to explore several passive reconnaissance techniques and deduce the properties of active network devices. The experimental results show that some properties are observable regardless of whether or not encryption is used. In particular, the acknowledgment response times vary due to differences in vendor firmware implementations.

Keywords: ITU-T G.9959, Z-Wave, vulnerabilities, wireless sniffing

1. Introduction

The prevalence of wireless connectivity in industrial control and sensor systems is increasing because it extends communications ranges at lower cost than wired alternatives. A 2011 survey of industrial control system operators reported that wireless networks were deployed in 43% of industrial control systems and numerous additional deployments were projected [3]. The survey respondents reported the use of IEEE 802.11, WirelessHART, Bluetooth and ZigBee as well as proprietary wireless systems [3]. An analysis of vulnerabilities

and suggestions for mitigation have been published for these protocols, with the exception of the proprietary systems [15].

The growing use of proprietary systems makes it necessary to analyze and discuss their security implications, especially when considering these systems for use in critical infrastructure assets. Of the numerous proprietary systems, wireless systems based on the ITU-T G.9959 recommendation, which specifies a short range narrow-band digital radio communications transceiver operating in the sub-GHz spectrum, have significant potential for growth in the critical infrastructure. The most common commercial instantiation of ITU-T G.9959 is Z-Wave, which is standardized and marketed by the Z-Wave Alliance. The Z-Wave Alliance comprises more than 300 companies and is actively working to increase the adoption of Z-Wave products around the globe.

IEEE 802.15.4 networks (e.g., WirelessHART and ZigBee) fulfill low-rate communications roles similar to those of Z-Wave in the critical infrastructure. Recent research efforts (see, e.g., [4, 12, 14]) have proposed novel security strategies for these networks. However, Z-Wave product development has been significantly restricted by nondisclosure and confidentiality agreements that stifle open-source security and resilience research. As a result, the security implications of the use of Z-Wave networks in the critical infrastructure are not well understood. In order to address this issue, this chapter introduces open-source techniques and tools for evaluating the security of ITU-T G.9959 wireless networks (including Z-Wave products) used in critical infrastructure assets.

2. ITU-T G.9959-Based Z-Wave Protocol

The ITU-T G.9959 recommendation specifies the physical (PHY) and media access control (MAC) layers for short-range, narrow-band digital radio communications transceivers. All manufactures adhere to the PHY/MAC specifications to ensure interoperability, but market their devices based on the network and application layers.

ITU-T G.9959-based networks operate in unlicensed frequency bands (e.g., 908.4 MHz in North America, 860.4 MHz in Europe and additional frequencies in other regions). Data rates of 9.6 Kbps (Rate 1), 40 Kbps (Rate 2) and 100 Kbps (Rate 3) are supported, depending on the transceiver type. Networks have two basic types of nodes: (i) control nodes; and (ii) end device nodes. Control nodes initiate commands while end device nodes respond to commands. Leveraging mesh topologies, end device nodes also forward commands to other nodes that are not directly reachable by a control node. The protocol allows a maximum of four hops between nodes and a maximum of 232 nodes in one network.

Each Z-Wave network is identified by a unique 32-bit Home ID, which is programmed by the manufacturer on each control node. The Home ID allows multiple networks to operate in close proximity without overlap. While a network may contain multiple control nodes, only one control node may be designated as the primary controller and its Home ID uniquely identifies the

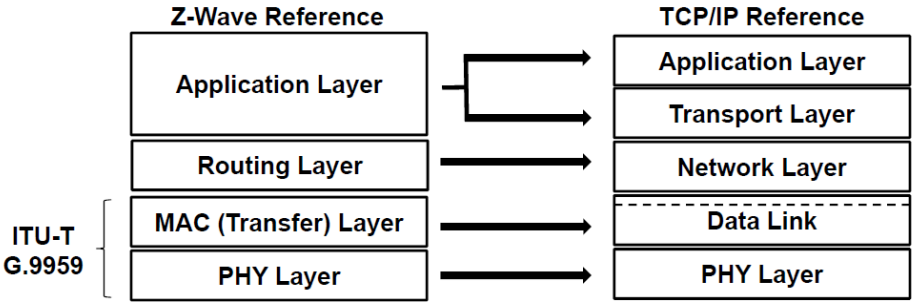


Figure 1. ITU-T G.9959/Z-Wave model mapped to the TCP/IP reference model.

network. During the network inclusion process, the primary control node assigns the new node an 8-bit Node ID, which is only unique in the local network.

The Z-Wave protocol consists of four layers. Figure 1 shows the four-layer ITU-T G.9959/Z-Wave reference model mapped to the five-layer TCP/IP reference model. The PHY layer controls access to the radio frequency medium, the MAC layer handles transmission and reception of frames between adjacent nodes, the routing layer controls the flow of messages throughout the mesh and the application layer executes commands associated with the end device.

2.1 PHY Layer

The PHY layer uses carrier sense multiple access with collision avoidance to control access to the wireless medium. As mentioned above, Z-Wave utilizes unlicensed frequency bands, which differ according to the region. The protocol offers three data rates: (i) 9.6 Kbps using frequency-shift keying with Manchester encoding; (ii) 40 Kbps using frequency-shift keying with non-return-to-zero encoding; and (iii) 100 Kbps using Gaussian frequency-shift keying with non-return to zero encoding.

As shown in Figure 2, the PHY protocol data unit (PPDU) consists of three main parts. The frame begins with a start header (SHR), which contains a preamble for symbol and bit synchronization, followed by a start of frame delimiter (SFD). The frame payload or PHY service data unit (PSDU) follows. Finally, for 9.6 Kbps data rate transmissions only, the frame concludes with an end header (EHR).

2.2 MAC Layer

The MAC layer (sometimes referred to as the transfer layer) is also detailed by ITU-T G.9959. The layer controls the transfer of data between two nodes and is responsible for frame acknowledgment, retransmission, data validation and notifying battery-operated devices to stay awake pending incoming transmissions.

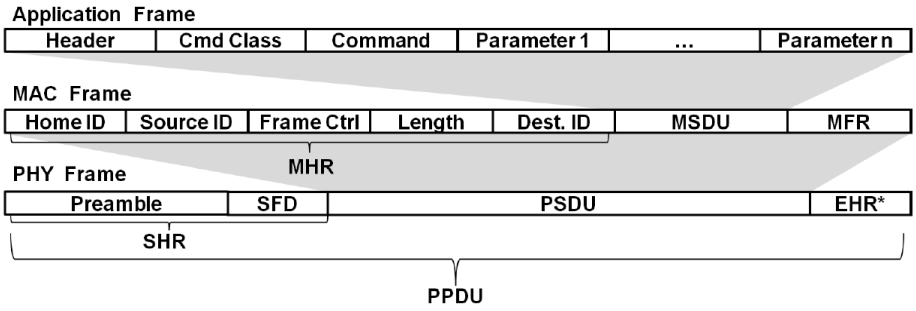


Figure 2. ITU-T G.9959 and Z-Wave frame formats.

There are three basic types of MAC frames: singlecast, acknowledgment and multicast. Each frame type follows the same general layout shown in Figure 2 with a MAC header (MHR), MAC service data unit (MSDU) and a MAC footer (MFR). Singlecast frames are transmitted to only one destination address (including the broadcast address). Acknowledgement frames are structured identically to singlecast frames, but have a MAC service data unit of zero length. Acknowledgement frames are sent in response to singlecast frames. Retransmissions occur when the sending node does not receive an acknowledgment from the receiving node. Multicast frames are sent to multiple destination nodes without acknowledgments.

The MAC header contains the Home ID, Source ID, frame control field, frame length and Destination ID (or bitmask in the case of a multicast frame). The MAC footer contains either an 8-bit checksum or a 16-bit cyclic redundancy check depending on the data rate used.

2.3 Routing Layer

Z-Wave mesh network topologies are managed by the routing layer to ensure that messages are successfully routed among control and end device nodes. The protocol specifies a maximum of 232 nodes and only one primary control node, although multiple secondary control nodes may exist in order to partition a network. Every node, with the exception of battery-operated devices, participates in routing by forwarding frames between nodes outside of direct wireless transmission range. The protocol also specifies a maximum of four hops between the primary control node and any other node.

The routing layer is responsible for scanning the network topology and maintaining a routing table in the primary control node. The routing table is built by the primary control node based on information received, upon inclusion or request, from each end device node about the neighbors of each node. The Z-Wave protocol stack supports automatic topology discovery and healing to optimize routing tables when the location of a node has changed or a node has been removed from the network (exclusion).

2.4 Application Layer

The majority of the application layer is implementation-specific depending on the Z-Wave developer. For brevity, this section discusses only what is applicable to all Z-Wave devices.

The application layer frame consists of the header, command class information and command parameters (see Figure 2). The application layer is responsible for executing the commands passed to it. Commands are broken into two classes: command class and device class.

A command class is related to a specific function or device. An example is the binary switchcommand class. The binary switch uses three commands: (i) `SET` to turn a device on or off; (ii) `GET` to request the status of a device; and (iii) `REPORT` to respond to the request. These three commands are foundational to all Z-Wave devices.

The device class is subdivided into the basic, generic and special device classes. The basic device class distinguishes between controllers, end devices and end devices that are capable of routing. The generic device class defines the function that the device performs as a controller or end device. The special device class allows for more specificity in device functionality.

It is important to note that the Z-Wave protocol supports encryption using the Advanced Encryption Standard (AES) with 128-bit keys. When implemented, the application frame is encrypted and an 8-byte authentication frame header is appended to the end of the MAC service data unit. While data encryption is supported by the protocol, its implementation is left to the manufacturer to decide if the device transmission is sensitive enough to warrant encryption. In wireless sensor networks, memory and power are scarce resources, which discourages developers from implementing encryption "unless required" [6]. Surveys of similar low-rate networks have demonstrated that the use of encryption or other security measures are far from ubiquitous [13].

3. ITU-T G.9959/Z-Wave Attack Classes

While Z-Wave is a proprietary protocol, vendors may purchase software development kits to produce Z-Wave certified products. Given the nature of hardware and software development, certain vulnerabilities are introduced by developer-specific implementation faults as illustrated in [5]. This section focuses on vulnerabilities of the underlying ITU-T G.9959 recommendation, which are common to all devices and attack classes that exploit the vulnerabilities.

Three classes of attacks are considered: (i) reconnaissance; (ii) denial-of-service; and (iii) packet injection. The three classes of attacks undermine network confidentiality, availability and integrity, respectively. A reconnaissance attack involves the passive collection of traffic or the active probing of a target network to gain information without interfering with normal operations. A denial-of-service attack prevents wireless system access and causes varying

degrees of system unavailability. A packet injection attack involves the transmission of specially-crafted packets to manipulate network or device behavior.

3.1 Reconnaissance Attacks

Reconnaissance lays the foundation for sophisticated follow-up attacks. The information acquired includes the protocols in use, device types, traffic flow patterns and even encryption keys if they are not handled properly. Information received from reconnaissance can help an attacker obtain accurate mappings of a system, services and/or vulnerabilities, enabling more significant attacks to be conducted in the future. Using a high-gain antenna, observations can be made at long distances, allowing an attacker to remain inconspicuous while gathering information. Apa and Hollman [1] have presented a proven exploitation of a wireless sensor network used in critical infrastructure assets. In particular, they demonstrated the exploitation of three devices from a maximum distance of 64 km.

An attacker armed with a directional antenna can capture ITU-T G.9959 transmissions for further analysis. Information gathered that might be useful to an attacker includes: (i) traffic patterns; (ii) use of encryption during transmission; and (iii) frame header content, which includes the unique Home ID of an ITU-T G.9959 network, Source ID of the device being sniffed and Destination ID.

Two demonstrations of ITU-T G.9959 exploitation have been published to date, both of them relied heavily on the ability to conduct reconnaissance to gain the knowledge required to craft follow-up attacks. Fouladi and Ghanoun [5] showed how to obtain a detailed understanding of the manner in which a secure door lock implements encryption and authentication; using this knowledge, they were able to discover a flaw that could be exploited. Picod et al. [11] were able to discern the specific commands and associated bit values that could be used to turn an alarm on and off. Follow-on attacks of these example attacks are discussed later in this chapter.

Several entities have developed sniffers for intercepting and transmitting Z-Wave frames. Sigma Designs [17] markets a closed-source development kit for Z-Wave device developers. The kit incorporates several hardware development platforms, technical documentation, software tools and a Z-Wave protocol sniffer. However, the kit comes with a non-disclosure agreement, which restricts the use of the tools.

A second sniffer project is z-force [5]. The sniffer includes custom firmware hosted on a CC1110 development board [20] and a z-force personal computer application. At the time of this writing, z-force has only been demonstrated on European Z-Wave frequency bands. The closed-source nature of z-force makes it difficult to evaluate and extend.

A third Z-Wave sniffer is Scapy-Radio, part of the open-source `hackrf` project [11]. Scapy-Radio integrates Scapy, a Python environment for manipulating network traffic and GNU Radio, an open-source signal processing toolbox. The tool includes GNU Radio companion implementations of Z-Wave,

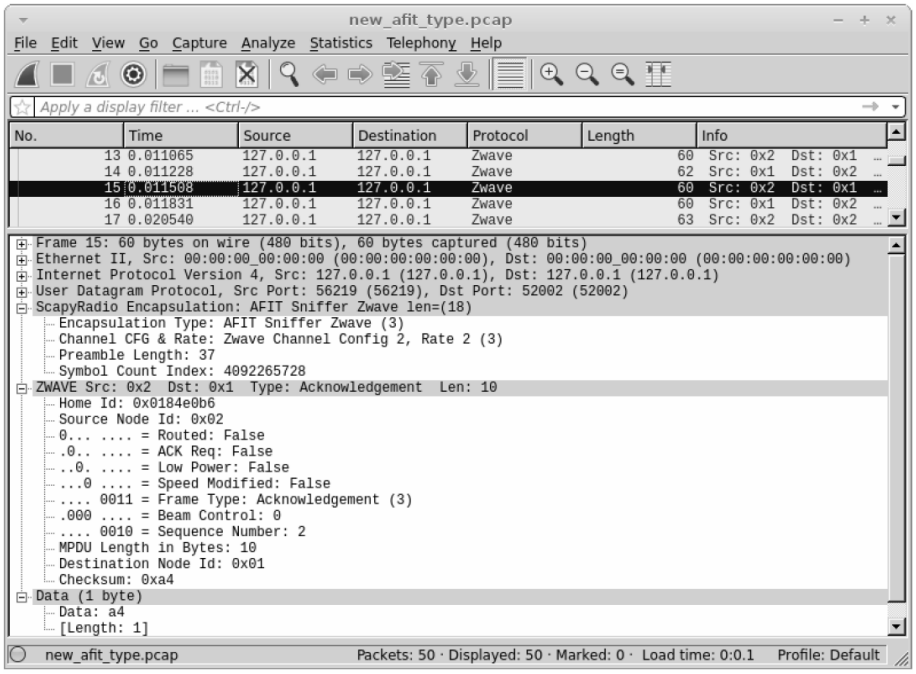


Figure 3. Wireshark Z-Wave dissection.

Bluetooth and ZigBee transceivers. The Z-Wave implementation receives samples from a software-defined radio, demodulates, synchronizes and decodes NRZ or Manchester encodings. As with z-force, the implementation is tuned to European bands, but can be modified for other bands using GNU Radio tools.

Application layer sniffers are available from OpenZWave. OpenZWave is an open-source project that provides libraries and drivers to communicate with USB-based Z-Wave controllers (e.g., Z-Stick S2 from Aeon Labs). OpenZWave provides a network querying tool to demonstrate its MinOZW library API. The `python-openzwave` package, an open-source wrapper for OpenZWave, provides a Python version of MinOZW and a Z-Wave shell interface to interact with a Z-Wave controller.

While Scapy or a `netcat` listener could be used to collect the encapsulated UDP frames sent over a local loopback device, future work will support the capture and analysis of Z-Wave frames using Wireshark. Wireshark is capable of intercepting UDP datagrams, but is unable to decode the encapsulation header, MAC header or Z-Wave payloads.

As part of this research, a packet dissector was developed for the encapsulation header and Z-Wave MAC header. Figure 3 shows a Z-Wave frame dissection. The dissector also decodes the first byte of the PHY service data unit payload to identify the application payload command class. The remaining payload bytes depend on the value of the command class field. The dissector

Table 1. Jamming efficiency against ITU-T G.9959 frames.

Integrity Data Rate	Check*	Max Payload	Jammer Bits to Jam	Efficiency
9.6 Kbps (Rate 1)	8-bit checksum	512 bits	1	512
40 Kbps (Rate 2)	8-bit checksum	512 bits	1	512
100 Kbps (Rate 3)	16-bit CRC	1,360 bits	1	1,360

*ITU-T G.9959 uses non-correcting integrity checks.

computes a checksum for each frame and denotes the outcome to the user in the info field of the packet list window. Dissectors for each of the 49 command classes, as identified in the OpenZWave source code, are currently being developed.

3.2 Denial-of-Service Attacks

Denial-of-service attacks prevent or degrade legitimate access to resources (e.g., radio frequency spectrum). Such attacks can be easily accomplished using off-the-shelf equipment [10]. Denial-of-service attacks on wireless networks can deny or degrade access to resources from the physical layer to the network layer. For example, the physical layer may be susceptible to narrow-band jamming, the carrier sense algorithms may be exploited to deny access to the medium [22] and routers can be consumed with overflowing interface queues.

Constant and deceptive jamming are effective for conducting denial-of-service attacks due to the MAC layer collision avoidance characteristics of ITU-T G.9959. When an attacker is operating a constant or deceptive jammer, any node within range will sense the channel as busy and wait to transmit. Even more effective and efficient is reactive jamming [8], which is difficult to detect [18]. A reactive jammer, that only transmits after the preamble and start of a frame delimiter of an ITU-T G.9959 PHY frame are detected, merely has to corrupt one bit of the PHY service data unit in order to cause an integrity check error and the complete loss of the frame. The non-correcting integrity checks used by Z-Wave are capable of detecting, but not correcting, single-bit errors. Even worse, the corruption of a single bit in the Z-Wave PHY layer, unlike a PHY layer that uses a spreading technique such as direct-sequence spread spectrum (DSSS), is achievable using narrow-band jamming. The use of error correction codes, as in IEEE 802.11a, is more robust to bitwise jamming [9]. Table 1 presents an estimate of jammer efficiency in terms of bits jammed per bit transmitted against ITU-T G.9959 (i.e., ratio of communications effort to jammer effort) based on the results in [9].

Depending on the objective, an attacker may use any of the methods described above to impact the availability of one or more nodes in a wireless network. For example, a Z-Wave network containing a thermostat (sensor) and water valve (actuator) could be subjected to a denial-of-service attack that

prevents the thermostat from reporting the current temperature or obstructs a command to activate the water valve. A similar scenario was successfully demonstrated in [15] with a gas pipeline remote terminal unit that included a wireless pressure sensor, pump and relief valve.

3.3 Packet Injection Attacks

Due to the broadcast nature of wireless networks, an attacker armed with information gained via reconnaissance may be able to inject forged packets into a network. The ability to conduct packet injection enables the attacker to masquerade as a legitimate network device while transmitting messages to manipulate system operation. Badenhop and Mullins [2] have investigated network degradation attacks against wireless routing protocols similar to those used in Z-Wave.

Using publicly-available hardware and software, researchers have reported the ability to conduct packet injection attacks to manipulate ITU-T G.9959 devices. Fouladi and Ghanoun [5] have developed a packet inspection/injection tool using a Texas Instruments radio transceiver and custom software (now publicly-available, albeit not open source). Using the tool, Fouladi and Ghanoun were able to inject traffic to exploit a vulnerability in a device-specific encryption implementation at the application layer that enabled them to send encrypted commands to perform unauthorized actions.

Picod et al. [11] have also demonstrated a packet injection attack against ITU-T G.9959 devices. Their packet inspection/injection tool uses a software-defined radio and open-source software packages. They demonstrated the ability to sense legitimate `SWITCH_BINARY_ON` commands and automatically inject subsequent `SWITCH_BINARY_OFF` commands, which effectively nullified legitimate messages.

These preliminary case studies demonstrate the attack possibilities when a malicious entity has the ability to inject forged ITU-T G.9959 traffic. In addition to these examples, an attacker may be able to: (i) flood a network with traffic causing a denial of service at the routing or application layers; (ii) send false status messages; or (iii) provide a control node with false routing information to poison the network routing table.

If the target ITU-T G.9959-based network is used in a critical infrastructure asset, the ability to inject commands and report false state information could prove disastrous. For example, an attacker may choose to send a close command to a water valve in a cooling system immediately following every open command sent by the master terminal unit while falsely reporting to the human-machine interface that the valve is open.

4. Passive Reconnaissance Techniques

Several high-level attack classes for Z-Wave systems have been presented above. A basic, invariant system threat from which other threats originate is passive reconnaissance. In the remainder of this chapter, the focus is on

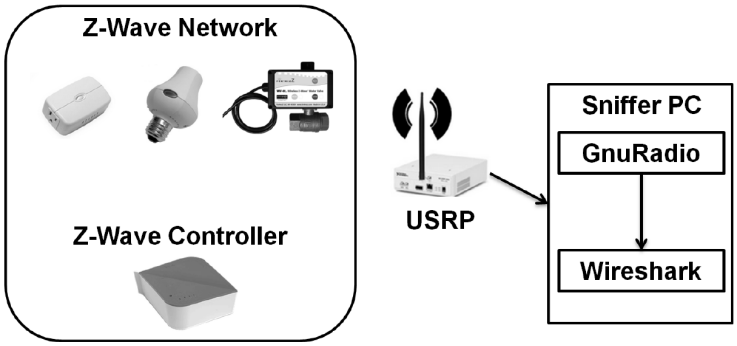


Figure 4. Sniffer architecture.

identifying several low-complexity passive reconnaissance techniques to further the exploration of Z-Wave security.

To capture arbitrary Z-Wave frames, a receiver chain was constructed using a USRP N210, GNU Radio, Scapy-Radio and a Z-Wave Wireshark packet dissector. Figure 4 shows the sniffer architecture. The USRP was tuned to a center frequency of 908.40 MHz to collect frequency-shift-keyed symbols at ±20 KHz deviation from the center frequency. An Ettus VERT900 dipole antenna was connected directly to the USRP and sampling was performed at 800 Kbps with 20 MHz bandwidth. Filtering, demodulation and symbol synchronization were performed by GNU Radio. Scapy-Radio, a subsystem within GNU Radio, provided preamble detection, byte synchronization, NRZ decoding and frame extraction operations. The extracted frames were transmitted over the `localhost` interface encapsulated as UDP datagrams to port 52002. Wireshark captured traffic over the `localhost` interface to intercept the datagrams. A custom Z-Wave packet dissector was used to decode the encapsulated Z-Wave frames, which were preserved in the PCAP format for later analysis. The experimental setup in Figure 4 permitted the passive collection of Z-Wave frames. Unlike application level sniffers, Z-Wave frames were captured regardless of the frame Home-ID. Moreover, the PHY and MAC layers were retained rather than stripped in order to provide insights into system behavior.

4.1 Controller-Device Pairing

Observations of the responses of the included device to the controller revealed significant details about the hardware, software and current state. At the end of an inclusion process, the controller interrogates the included device to learn about it. This information is used by the controller to present accurate control options and device information to the user.

Figure 5 shows a frame capture of an Aeon Labs Z-Stick controller using a Jasco JS-45603 smart dimmer. The data in the application payloads, which was reversed by referencing the Open-ZWave command class source code, is summarized to the right of Figure 5. The figure shows a sequence of `GET` commands

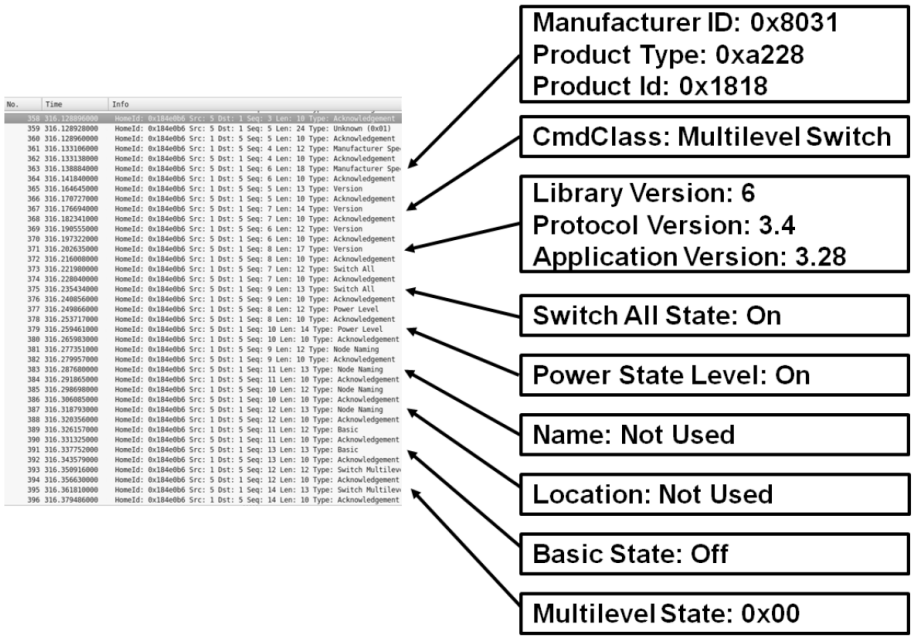


Figure 5. Pairing between a Jasco smart dimmer and an Aeon Z-Stick controller.

from the controller to the dimmer. The dimmer responds to each request with a `REPORT` message that reveals information. Device-level information, such as the vendor, device type and ID are provided in a manufacturer-specific `REPORT` message. The controller requests software version information, to which the dimmer device replies with a Z-Wave library, protocol and application version. After the device and software level information is gathered, the controller queries the device on the current state of each configuration and control item. This state information can be used by an observer to determine the set of messages that the device services, which is especially useful when the device-specific data refers to an unknown product type or manufacturer ID. The location state is of particular interest. While location information was not initialized in the capture shown in Figure 5, it is still possible for a device to reveal some information about its configured location. The location information is in the form of a grouping index that corresponds to a user-defined label assigned at the controller interface. A user may group devices by room, floor or functionality, so additional information is required to understand the semantics of this value.

If the pairing is not observable, passive reconnaissance may still capture the interactions between the controller and end devices. As with the latter portion of the pairing process, the command classes of observed messages provide information about the device function. The drawback is that the observation period is significantly longer than if observed during the pairing process. This is especially true for low-power event triggered devices. These devices operate

No.	Time	Protocol	Length	Info
52	2.344342000	Zwave	63	HomeId: 0x184e0b6 Src: 1 Dst: 5 Seq: 8 Len: 13 Type: Switch Multilevel
54	2.353639000	Zwave	60	HomeId: 0x184e0b6 Src: 5 Dst: 1 Seq: 8 Len: 10 Type: Acknowledgement
56	2.366168000	Zwave	62	HomeId: 0x184e0b6 Src: 1 Dst: 5 Seq: 9 Len: 12 Type: Switch Multilevel
58	2.368858000	Zwave	60	HomeId: 0x184e0b6 Src: 5 Dst: 1 Seq: 9 Len: 10 Type: Acknowledgement
60	2.375081000	Zwave	63	HomeId: 0x184e0b6 Src: 5 Dst: 1 Seq: 6 Len: 13 Type: Switch Multilevel
62	2.380123000	Zwave	60	HomeId: 0x184e0b6 Src: 1 Dst: 5 Seq: 6 Len: 10 Type: Acknowledgement

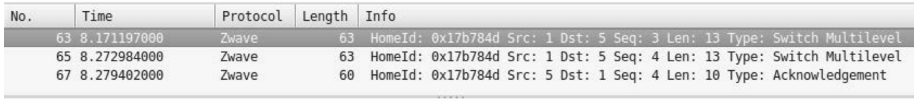
Figure 6. Z-Stick controller `SET` command capture.

at low duty cycles to extend battery life and only emit frames when a specific monitored event occurs (e.g., a door opening or an excessive sensor reading).

4.2 Controller-Specific Behavior

A controller may exhibit discernible differences in how it interacts with devices. To illustrate this observation, the dimmer (multilevel switch class device) was paired with two different controllers to show that the frames are noticeably different.

In the first experiment, the Jasco Smart Dimmer was paired with an Aeon Labs Z-Stick S2 controller (Node 1). The multilevel switch `SET` command was executed through an OpenZWave control panel while the Z-Wave frames were captured by the Scapy-Radio USRP sniffer and processed by the Wireshark Z-Wave dissector. Note that the Z-Stick controller has an ID of 1 and the dimmer has a node ID of 5. The Z-Stick controller sends a multilevel dimmer command to set the dimmer to a value, shown as frame 52, which is acknowledged by the device. After receiving the acknowledgment (ACK), the controller requests an update of the current dimmer state (output voltage) by sending frame number 56 in Figure 6. The dimmer complies by sending its current value to the controller in frame 60 in Figure 6. This results in six frames being exchanged per transaction. The same pattern occurs when the Aeon Z-Stick controller sends commands to other devices. Essentially, the controller issues a `SET` command, followed by a `GET` command on the same value to confirm that the device complied with the command. The targeted device responds to the command with a `REPORT` command.



No.	Time	Protocol	Length	Info
63	8.171197000	Zwave	63	HomeId: 0x17b784d Src: 1 Dst: 5 Seq: 3 Len: 13 Type: Switch Multilevel
65	8.272984000	Zwave	63	HomeId: 0x17b784d Src: 1 Dst: 5 Seq: 4 Len: 13 Type: Switch Multilevel
67	8.279402000	Zwave	60	HomeId: 0x17b784d Src: 5 Dst: 1 Seq: 4 Len: 10 Type: Acknowledgement

Figure 7. VeraLite controller `SET` command capture.

In the second experiment, the same dimmer device was paired with a VeraLite controller. Using the VeraLite controller web interface, the multilevel switch `SET` command was invoked while capturing Z-Wave frames. Figure 7 shows the results of the capture. Coincidentally, the node IDs are once again one and five. The VeraLite controller sends two versions of the same multi-

level switch `SET` command. In frame 63, the ACK-required bit in the frame control field is not set; however, it is set in frame 65. After the dimmer device receives frame 65, it acknowledges receipt to the controller, which is shown in Figure 7 by the matching sequence numbers. The controller initiates the same command with and without the ACK-required bit set is very likely because the controller is compatible with devices that are not able to reply. Devices that are unable to transmit may ignore frames that require an ACK. As with the Z-Stick controller, this pattern is observable when the VeraLite controller issues `SET` commands to devices. The `SET` command is sent both with and without the ACK and does not necessarily expect a reply.

The results show that there is more than one way to execute a switch `SET` transaction. While it cannot be concluded that the observed patterns are unique to the controllers, it can be deduced that a given command pattern reduces the possible identities of the target controller. For example, when an unknown controller exhibits a pattern consistent with that of a Z-Stick controller, this would imply that the unknown device is not a VeraLite controller.

4.3 Device-Specific ACK Times

When the application layer of Z-Wave frames are encrypted, passive reconnaissance of a device using the previously-identified methods may prove ineffective. Application layer encryption obscures the command class fields, making it more difficult to identify controller-specific command patterns. An outside observer must turn to non-encrypted observations to deduce properties of the devices. Implementation differences between hardware, software and the physical environments of devices may result in observable differences outside of the encrypted traffic. Existing techniques such as traffic analysis and side channel analysis have been shown to be effective at thwarting confidentiality mechanisms like encryption [21].

A fingerprinting technique known as preamble manipulation may be applicable to Z-Wave devices [14]. Unfortunately, it was difficult to find a Z-Wave device that uses a transceiver other than the ZM3102N to verify this technique. Since the experimental hardware was homogeneous, firmware implementations could be examined behaviorally to identify implementation and functionality differences. As a matter of fact, the 32 KB of flash memory available in the ZM3102N provides ample opportunities for vendors to customize their implementations.

One ubiquitous observable behavior below the application layer is the time taken by a node to send an ACK upon receiving a request. Time-of-flight fingerprinting metrics are explored in [16], where the authors report that measurement variance is due to hardware and environmental factors. In their experiments, laptop computers were used to perform packet injection, ACK response and time measurements. Each had an operating system that managed independent operations, but also competed for CPU clock cycles, resulting in variations in the time-of-flight measurements. While this is a valid issue, Z-Wave devices are far more specialized than laptop computers, having application-specific I/O

and fewer, if any, concurrent tasks. This suggests that an ACK response of a Z-Wave device is more deterministic than a general purpose laptop. If this is true, ACK-based fingerprinting would be a valid approach for identifying Z-Wave devices.

The experiments described in the remainder of this section demonstrate that ACK-based fingerprinting of Z-Wave devices shows promise. While the Z-Wave devices used in the experiments were all embedded systems, the sniffer ran on a Dell Latitude E6520 laptop. The laptop incurred a degree of the dynamics identified in [16]. The ACK delay time was measured using the packet arrival times observed by the Wireshark Z-Wave dissector. The time values were determined by the Wireshark process when an encapsulated Z-Wave frame was observed on the `localhost` interface. Encapsulated frames were sent over the `localhost` interface by the GNU Radio process. The GNU Radio process received samples from the USRP FPGA via a gigabit Ethernet interface. In the experiments, each process, interface and device driver contended with other resources on the sniffer laptop and were subject to contention dynamics. Regardless of the dynamics, adequate sampling of ACK delays eliminated the confounding effects from the measurements. Seventy ACK response times were collected for each device under test to mitigate sampling bias and to facilitate observational comparisons.

ACK Times for Different Vendor Devices. In the first experiment, a VeraLite controller was used to repeatedly issue commands to a FortrezZ water valve. Each command generated several acknowledged frames from the controller and valve. The ACK response to a particular singlecast frame could be identified by the matching four-bit sequence numbers. After carefully pairing the singlecast frames with their associated ACKs, the response time was measured as the difference in time of between the observation of the singlecast frame being sent and the observation of the ACK being sent. The distances between the water valve, VeraLite controller and the Z-Wave sniffer were all within one meter of one another, implying that signal propagation delays were negligible.

Figure 8 shows the 99% confidence intervals estimating the true mean ($n = 70$) of the ACK response times for the VeraLite controller and FortrezZ water valve. The mean response times are different with a two-sample t-test p-value of 0.00; clearly, the response time of the water valve is greater than that of the VeraLite controller. Moreover, the difference in the response times is an order of magnitude more than variations due to thermal effects [23]. Since both devices use the same hardware, the differences may be due to firmware implementation differences.

ACK Times for Same Vendor Devices. The same experiment was repeated, except that the selected controller and device originated from the same vendor. The Aeon Labs Z-Stick S2 controller was used as the controller and an Aeon Labs Appliance Switch was used as the device. A total of 70

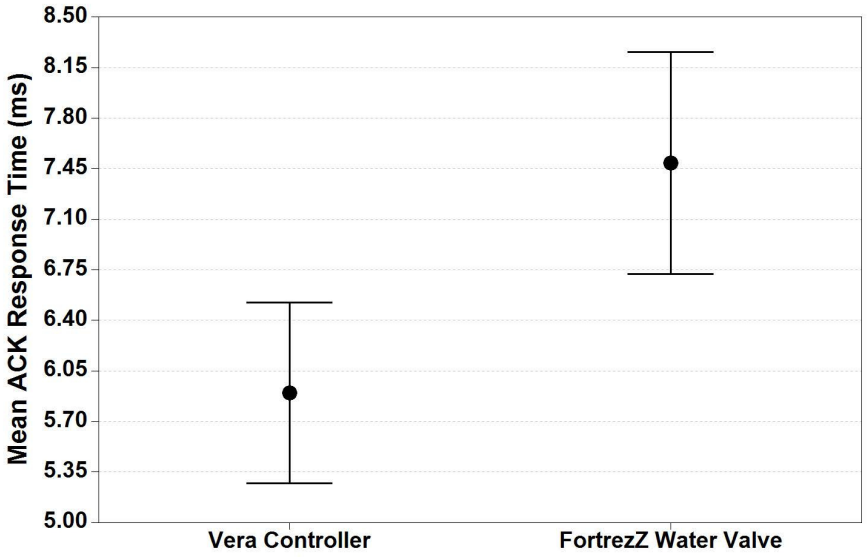


Figure 8. Mean ACK response times for devices from two different vendors.

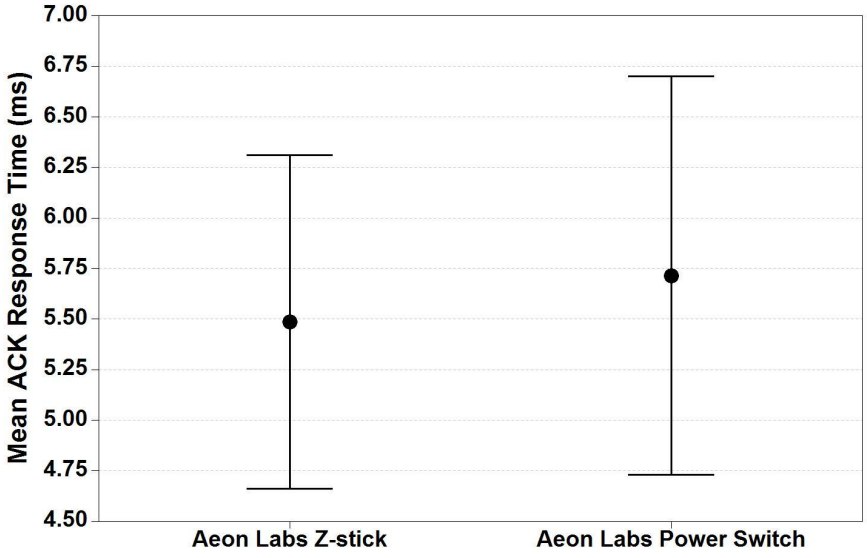


Figure 9. Mean ACK response times for devices from the same vendor.

ACKs were recorded for both systems by repeatedly issuing commands from the controller. Figure 9 shows the observed mean ACK response times for the two devices ($n = 70$ and a confidence interval of 99%). In this case, the mean ACK response times of the two Aeon Labs devices are not statistically

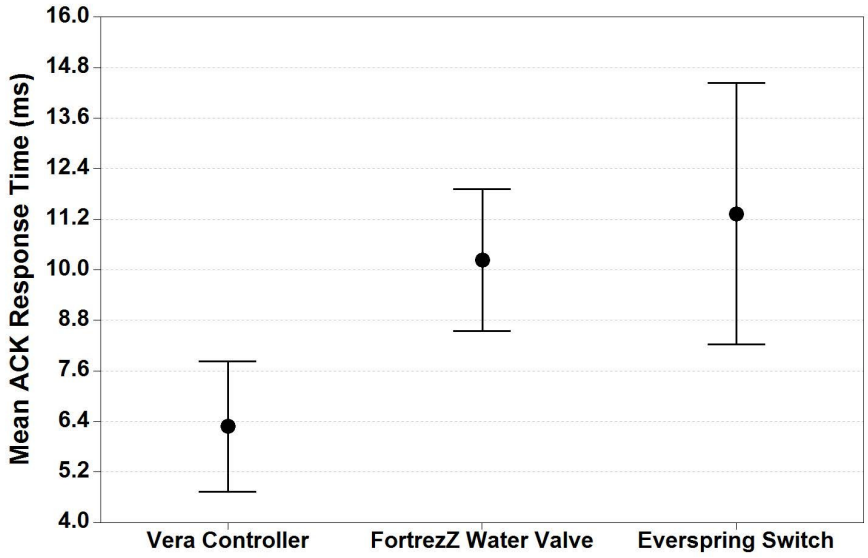


Figure 10. Mean ACK response times for a 60-second polling interval.

different (two-sample t-test p-value = 0.64). This suggests that both devices have similar ZM3102N implementations.

ACK Times for Polling Commands. In this experiment, instead of manually creating Z-Wave traffic, the VeraLite controller was configured to automatically poll two devices every 60 seconds. The two devices polled were the FortrezZ WV01LFUS075 water valve and an Everspring AN145 lamp socket switch. Figure 10 shows the mean response delays for each device with 99% confidence intervals ($n = 70$). The figure reveals several interesting points. First, there is a statistically significant difference between the response times of the VeraLite controller and the other two devices (ANOVA p-value = 0.00). The difference of means between the water valve and lamp switch is not statistically significant, but the variance is higher for the Everspring switch. Second, the mean ACK response time of the water valve is different from that seen in Figure 8. The variances of the devices in Figure 10 are larger than those reported in Figures 8 and 9. This experiment had 60 seconds between each transaction, whereas the two previous experiments had repetitions at intervals of one or two seconds. During the 60-second period of inactivity, it is hypothesized that the devices went into a lower power state, which increased their ACK response times.

Differences in Mean ACK Response Times. The experimental data suggests that the differences in means are due to implementation differences in how ACKs are handled by the devices. Upon receiving a frame, a checksum is

performed. If the checksum passes and the frame has either requested an ACK or the protocol requires it, the recipient generates an ACK using the sequence number of the received frame and sends the response to the originator. The differences arise depending on when these steps are taken during the transceiver chains. The ACK response may be initiated immediately upon receiving the frame or after the frame is queued so that the receiver may quickly return to search for the next frame. This option, for example, is provided by the CC2420 transceiver [19].

After the ACK is generated, another implementation decision involves the transmission of the ACK frame. If the ACK is given priority, it is either placed at the front of the send queue or an interrupt is generated to force the transceiver to immediately transmit the ACK. Other design choices include the sizes of the receive and send queues, buffer exception handling and the queue service rates. In a delay-prone scenario, ACKs are handled by an attached microprocessor via the SPI bus.

Another source of differences may be the ZM3102N configuration settings. The ITU-T G.9959 specification lists parameters relevant to ACK response times such as *MacMinAckWaitDuration*, *TurnaroundTimeRXTX* and *MacMin CCARetryDuration*. The three parameters specify the minimum frame spacing between receiving a packet and transmitting its ACK, the time penalty for switching from the receive mode to the transmit mode and the minimum time to wait between clear channel assessments, respectively. It is not clear which, if any, of these parameters are configurable on the ZM3102N; however, each may impact the ACK response time.

5. Conclusions

This chapter has examined the security implications of using ITU-T G.9959 wireless networks with Z-Wave devices in critical infrastructure assets. Several techniques for passively discriminating between Z-Wave devices based on functionality and vendor were investigated. Experiments involving passive observations of ACK response times demonstrate that it is possible to identify implementation differences in ZM3102N firmware. In particular, the experimental results reveal that a VeraLite controller and FortrezZ water valve have different mean ACK response times, while an Z-Stick controller and appliance switch from the same vendor (Aeon Labs) have equivalent response times. These results suggest intra-vendor similarities.

Future research will focus on developing new passive techniques and refining existing techniques for use with other types of devices. Additionally, research will attempt to understand the reasons for the differences in the mean ACK response times. Efforts will also explore active fingerprinting techniques. One such technique will involve sending messages corresponding to all the command classes to a target device to discern its capabilities. The ultimate goal is to develop a tool that leverages passive and active techniques in device fingerprinting, with functionality similar to the popular `nmap` operating system fingerprinting tool.

Note that the views expressed in this chapter are those of the authors and do not reflect the official policy or position of the U.S. Air Force, U.S. Army, U.S. Department of Defense or U.S. Government.

References

[1] L. Apa and C. Hollman, Compromising industrial facilities from 40 miles away, presented at the *Black Hat USA Conference*, 2013.

[2] C. Badenhop and B. Mullins, A black hole attack model using topology approximation for reactive ad-hoc routing protocols, *International Journal of Security and Networks*, vol. 9(2), pp. 63–77, 2014.

[3] W. Boyes, All quiet on the wireless front, Control Global, Schaumburg, Illinois (`www.controlglobal.com/articles/2011/all-quite-on-the-wireless-front`), August 9, 2011.

[4] C. Dubendorfer, B. Ramsey and M. Temple, ZigBee device verification for securing industrial control and building automation systems, in *Critical Infrastructure Protection VII*, J. Butts and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 47–62, 2013.

[5] B. Fouladi and S. Ghanoun, Security evaluation of the Z-Wave wireless protocol, presented at the *Black Hat USA Conference*, 2013.

[6] Freescale Semiconductor, Freescale Beestack: Application Development Guide, Document Number: BSADG Rev. 1.1, Chandler, Arizona, 2008.

[7] InGuardians, Converting Radio Signals to Data Packets: Examination of Using GNU Radio Companion for Security Research and Assessment, Washington, DC (`www.inguardians.com/pubs/GRC_signal_analysis_InGuardians_v1.pdf`), 2014.

[8] M. Li, I. Koutsopoulos and R. Poovendran, Optimal jamming attack strategies and network defense policies in wireless sensor networks, *IEEE Transactions on Mobile Computing*, vol. 9(8), pp. 1119–1133, 2010.

[9] G. Noubir, Robust wireless infrastructure against jamming attacks, in *Handbook on Securing Cyber-Physical Critical Infrastructure: Foundations and Challenges*, S. Das, K. Kant and N. Zhang (Eds.), Morgan Kaufmann, Waltham, Massachusetts, pp. 123–145, 2012.

[10] K. Pelechrinis, M. Iliofotou and S. Krishnamurthy, Denial-of-service attacks on wireless networks: The case of the jammers, *IEEE Communications Surveys and Tutorials*, vol. 13(2), pp. 245–257, 2011.

[11] J. Picod, A. Lebrun and J. Demay, Bringing software defined radio to the penetration testing community, presented at the *Black Hat USA Conference*, 2014.

[12] B. Ramsey and B. Mullins, Defensive rekeying strategies for physical-layer-monitored low-rate wireless personal area networks, in *Critical Infrastructure Protection VII*, J. Butts and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 63–79, 2013.

[13] B. Ramsey, B. Mullins, R. Speers and K. Batterton, Watching for weakness in wild WPANs, *Proceedings of the IEEE Military Communications Conference*, pp. 1404–1409, 2013.

[14] B. Ramsey, T. Stubbs, B. Mullins, M. Temple and M. Buckner, Wireless infrastructure protection using low-cost radio frequency fingerprinting receivers, *International Journal of Critical Infrastructure Protection*, vol. 8, pp. 27–39, 2015.

[15] B. Reaves and T. Morris, Analysis and mitigation of vulnerabilities in short-range wireless communications for industrial control systems, *International Journal of Critical Infrastructure Protection*, vol. 5(3-4), pp. 154–174, 2012.

[16] L. Schauer, F. Dorfmeister and M. Maier, Potentials and limitations of WiFi-positioning using time-of-flight, *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation*, 2013.

[17] Sigma Designs, Z-Wave Products, Fremont, California (`z-wave.sigmadesigns.com/products`), 2015.

[18] M. Strasser, B. Danev and S. Capkun, Detection of reactive jamming in sensor networks, *ACM Transactions on Sensor Networks*, vol. 7(2), article no. 16, 2010.

[19] Texas Instruments, 2.4 GHz IEEE 802.15.4/ZigBee-Ready RF transceiver, CC2420, SWRS041c, Dallas, Texas (`www.ti.com/lit/ds/symlink/cc2420.pdf`), 2014.

[20] Texas Instruments, Low-Power Sub-1 GHz RF Transceiver, CC1101, SWRS0611, Dallas, Texas (`www.ti.com/lit/ds/symlink/cc1101.pdf`), 2015.

[21] C. Wright, L. Ballard, S. Coull, F. Monrose and G Masson, Uncovering spoken phrases in encrypted voice-over-IP conversations, *ACM Transactions on Information and System Security*, vol. 13(4), article no. 35, 2010.

[22] W. Xu, W. Trappe, Y. Zhang and T. Wood, The feasibility of launching and detecting jamming attacks in wireless networks, *Proceedings of the Sixth ACM International Symposium on Mobile Ad Hoc Neworking and Computing*, pp. 46–57, 2005.

[23] Zensys, ZM3102N Z-Wave Module Datasheet, Document No. DSH10756, version 6, Fremont, California (`media.digikey.com/pdf/Data%20Sheets/Zensys%20PDFs/ZM3102N.pdf`), 2007.