TECHNICAL

WHITE PAPER

Compromising Industrial Facilities from 40 Miles Away

by Lucas Apa and Carlos Mario Penagos Hollman

Security Researchers, IOActive, Inc.

Abstract

The evolution of wireless technologies has allowed industrial automation and control systems (IACS) to become strategic assets for companies that rely on processing plants and facilities in industries such as energy production, oil, gas, water, utilities, refining, and petrochemical distribution and processing. Effective wireless sensor networks have enabled these companies to reduce implementation, maintenance, and equipment costs and enhance personal safety by enabling new topologies for remote monitoring and administration in hazardous locations.

However, the manner in which sensor networks handle and control cryptographic keys is very different from the way in which they are handled in traditional business networks. Sensor networks involve large numbers of sensor nodes with limited hardware capabilities, so the distribution and revocation of keys is not a trivial task.

In this paper, we review the most commonly implemented key distribution schemes, their weaknesses, and how vendors can more effectively align their designs with key distribution solutions. We also demonstrate some attacks that exploit key distribution vulnerabilities, which we recently discovered in every wireless device developed over the past few years by three leading industrial wireless automation solution providers. These devices are widely used by many energy, oil, water, nuclear, natural gas, and refined petroleum companies.

An untrusted user or group within a 40-mile range could read from and inject data into these devices using radio frequency (RF) transceivers. A remotely and wirelessly exploitable memory corruption bug could disable all the sensor nodes and forever shut down an entire facility. When sensors and transmitters are attacked, remote sensor measurements on which critical decisions are made can be modified. This can lead to unexpected, harmful, and dangerous consequences.

IOActive™

COMPREHENSIVE COMPUTER SECURITY SERVICES

Contents

Introduction

Industrial automation and control systems (IACS) are used in:

- Manufacturing
- Processing plants and facilities
- Utilities
- Building environmental control systems
- Oil, gas, and water, pipelines
- Petroleum production and distribution

Automation allows these organizations to carefully study and anticipate optimal responses to measured conditions and automatically execute these responses when the conditions occur. More importantly, automation allows organizations to increase productivity, improve critical machinery reliability, and minimize the risk of environmental disasters.

Recent wireless communication technologies have enabled the creation of new plant automation architectures that give organizations strategic advantages by replacing key pieces of their hard-wired infrastructure. These advantages include cost savings in logistics, installation, and engineering, as well as better data acquisition frequency and reliability. These wireless solutions also provide remote and localized control, allow for the efficient transmission of current and historical data to an organization's central office, and reduce the need to physically access potentially dangerous systems and machinery.

Due to the inherent risk associated with industrial machinery, an organization's highest priority is maintaining the availability and integrity of all system components. Business processes are directly impacted by IACS systems and devices. As a result, the interruption of information flowing to and from these devices could lead to the loss of trade secrets, violations of regulatory requirements, compromise and damage to public and employee safety, and the loss of life.

Most IACS sensor nodes have limited resources for handling network keys. Because they cost little, are slow, and use very little power, key distribution and management has been one of the most critical and important aspects of wireless sensor network security.

In this paper, we review a variety of standardized wireless sensor networks (WSNs). We also analyze vulnerabilities that we discovered and researched, which relate to wireless key distribution implementations. These vulnerabilities affect all devices produced over the last several years by three leading wireless automation device and system providers. Many leading oil, natural gas, and refined petroleum companies currently use these. These vulnerabilities could allow remote untrusted users or groups lacking inside access to remotely attack the sensors and measurements.

Research

In recent years, security researchers have become more interested in industrial control systems, because even the simplest of security bugs can lead to critical and frightening consequences. Most attack vectors being researched today are based on the assumption that an attacker is an “insider” and a trusted person, employee, or contractor with special access or information not generally known to the public. Despite much research effort, key distribution in wireless sensor network remains an unresolved problem.

Industrial Uses of Wireless Automation

Before wireless technologies emerged, copper and aluminum electrical wires were used to monitor and control remote instrument devices. Corrosion, creep resistance, ductility, and thermal conductivity are only some of the properties of these electrical conductors that must be taken into account and managed. Hard-wired systems are considered extremely reliable, because they are trusted and tangible. However, the cost of wires, trenching, mounting, and installation are clear disadvantages--even before taking into account environmental factors such as fire, galvanic corrosion and electrolysis.

Wireless solutions eliminate hard-wiring costs and the need for long distance analog I/O modules and analog-to-digital converters from the control instrumentation loop. Inputs and outputs can be relayed through Modbus or serial communication to and from the control device. Today critical applications such as failsafe control systems using wireless technology have also been integrated into wireless IACS devices and systems.

A variety of license-free frequencies (such as 5.8 Ghz, 2.4 Ghz, and 900 Mhz) are used with these devices. Each frequency has advantages and disadvantages relating to the bandwidth it requires and the distance that signals can travel. The carrier is equivalent to the cable, but it is the wireless protocol that determines the packet structure and handshaking mechanism.

Industrial companies do not use popular wireless network protocols such as IEEE 802.11 and Bluetooth. These more common wireless networking protocols were not designed with industrial facilities in mind. For example, consider all the electrical activity in an industrial company—such activity creates interference that disrupts network traffic.

Also, the prioritization of versatility and reliability over speed is not compatible with the main premise of the most common daily “business” protocols, which typically transmit much more data than industrial networks. Industrial control networks must be reliable, scalable, and adaptable.

Industries and Applications

In this section, we discuss some common industries and applications that can benefit from industrial wireless automation products--from basic monitoring and control to SCADA.

Oil and Gas

Oil and gas companies have turned to innovative technologies to meet increased worldwide demand for energy and to achieve operational goals. Operators must expand and find better ways to increase their current production levels while complying with environmental regulations. With technological advances in wireless communication and remote telemetry, oil and gas companies can automate and control production sites in widely dispersed geographic areas.

Upstream (production) applications:

- Plunger/artificial lift optimization
- Well-head automation
- RTU/EFM I/O extensions
- Cathodic protection monitoring
- Hydrogen sulfide (H2S) monitoring
- Pump/compressor station control systems

Midstream (pipeline/tank farm) applications:

- Tank level monitoring
- Pipeline cathodic protection
- Rectifier voltage monitoring
- Gas/liquid flow measurement
- Pipeline pressure and valve monitoring

Refining and Petrochemicals

For refining and petrochemical companies, flaring is an inevitable result of start-up and shut-down processes and a continued safety concern. Flaring often occurs when managing the disposal of waste gases in routine hydrocarbon operations. In order to accurately determine methane production levels and flows, sensors can detect both the pressure and vacuum levels present within the methane production system. In addition, sensors take temperature measurement readings at the point of heat in an active flame to verify that methane is burning.

Downstream (refining) applications:

- Raw material tank levels
- Flare temperature monitoring
- Remote terminal units (RTU) / electronic flow meters (EFM) IO extensions
- Pressure relief and shut-off valves
- Steam trap monitoring

- Flow meter monitoring

Utilities

Gas pipeline operators automatically collect and report corrosion prevention data and transmit it back to central SCADA systems through wireless RF networks. In the “old days”, technicians maintained these remote systems by physically going to a site to take readings and make adjustments. This expensive and time-consuming process has been replaced by industrial wireless devices. Below are some applications used in this industry:

Energy/utility applications:

- Transformer temperature
- Natural gas flow
- Power outage reporting
- Capacitor bank control
- kV, Amp, MW, MVAR reading

Industrial Process Monitoring and Control

Industrial companies that integrate wireless telemetry sensors into certain plant operations benefit not only from lowered implementation costs--they also benefit from optimized plant performance and productivity. The deployment of wireless devices into plant operations has translated into improved process efficiencies and optimization by measuring secondary process parameters.

Process monitoring and control applications:

- Liquid level measurement
- Valve/pump control
- Equipment (condition/status)
- Environmental monitoring

Water and Waste Water

Two or more remote facilities may be interconnected to enable wireless monitoring of process data and on-site control and execution. Control systems with remote wireless sensor and telemetry devices are ideal for the water industry. This is because water intake control logic can be optimized based on data collected from all delivery phases--including wells, pumps, tanks, and purification facilities.

Water and waste water applications:

- Lift station status
- Remote pumping stations
- Water treatment plants
- Water distribution systems
- Wastewater/sewer collection systems
- Water irrigation systems/agriculture

Key Distribution in Wireless Sensor Networks

Key distribution involves distributing secret keys between nodes to:

- Provide data confidentiality.
- Provide data integrity.
- Authenticate communicating entities.

In some key distribution schemes, cryptographic information is preloaded on every sensor node allowing them to communicate securely with one other on a network. This type of scheme must allow for additional network node creation after deployment.

Wireless sensor networks and traditional business networks handle and control cryptographic keys differently. Wireless sensor networks involve large numbers of sensor nodes and limited hardware capabilities.[3] In a basic key management system, many schemes exist for generating, ordering, distributing, storing, and destroying keys. However, many of these schemes suffer from high communication overhead and storage costs, low scalability, and poor resilience against different types of attacks.[1]

Challenges of Wireless Sensor Network Key Distribution

Wireless sensor networks are organized into distributed structures based on node capabilities. Base stations collect sensor readings and perform heavy operations on behalf of nodes and can act as gateways to other networks. In general, base stations are situated with cluster sensor nodes surrounding them to form a dense network.

Sensor network limitations complicate the design of secure protocols:[4]

- ***Deployment in public or hostile locations:*** Low-cost sensor nodes are not hardened against tampering, so physical attack is possible in public or hostile locations. Symmetric keys may be preloaded into the firmware of every device in a particular region. As a result, an attacker accessing the memory of a single device can obtain keys to access the entire network.
- ***Post-deployment knowledge:*** A security protocol should be keenly aware of which nodes are neighbors within a network. It is a costly and time-consuming task to pre-determine the location of individual nodes when a large number of nodes are being deployed.
- ***Limited bandwidth and transmission power:*** Typical sensor network platforms have limited bandwidth.

In addition to the general requirements (Availability, Authentication, Confidentiality, Integrity, and Non-repudiation) for securing wireless sensor networks, these are some specific requirements:

- ***Degradation:*** The ability to change security levels as resource availability changes.[6]

- ***Survivability:*** The ability to provide minimum security services even if there is a power loss, failure, or attack—if these events occur, the environment is considered malicious..

Security requirements also exist for implementing IACS key distribution mechanisms. Use the following metrics[5] to compare and evaluate IACS key distribution solutions:

- ***Efficiency:*** How much memory is required to store security credentials? How many processor cycles are required to establish a key? How many messages are exchanged during the key generation process?
- ***Scalability:*** Does the key distribution mechanism support large networks? Can it be scaled to support increases in the network size after deployment?
- ***Probability of key sharing:*** What are the odds that two or more sensor nodes will store the same key or keying material?
- ***Resilience:*** Is the solution designed to resist node capture or a compromise of security credentials?

Symmetric Key Distribution Protocols

These symmetric key distribution protocols were originally intended for traditional networks [3] but are suitable for wireless sensor networks:

- Preload a *single network key* onto all nodes prior to deployment. All communication with neighbor nodes is established using the shared key to encrypt messages and by appending a Message Authentication Code (MAC) to ensure integrity.
- Every node in the network issues a *unique shared key* that it shares with other nodes. Nodes must store an undetermined number of keys; as a result, they require a large memory storage capacity.
- The nodes authenticate to a *trusted base station* or *key distribution center (KDC)* where a link key is generated for both parties upon request. A single key is preloaded on each node. If an attacker captures a single node, information about the rest of the network will not be revealed to him. However, if the attacker compromises the trusted base station or KDC, the underlying security will be completely broken.

One IACS key distribution security requirement is Resilience against node capture. To meet this requirement, compromised security credentials should never reveal information about the security of any other links within the wireless sensor network.

Other key distribution schemes use certificates, asymmetric keys, and public key infrastructures to validate the authenticity of another device’s certificate.

The Journey of Radio Encryption Keys

When industrial device manufacturers use radio modules from other vendors, specific encryption keys are available to them for providing security. Device manufacturers can configure the radio modules with a custom key to automatically encrypt or decrypt all packages after processing. This key is stored on the radio module itself and nowhere else on the device board. Therefore, if this is the only mechanism your company uses to provide confidentiality, take note of its weaknesses and the types of attacks to which it is vulnerable.

A radio module's journey ends with the final user, but it is the device manufacturer who assembles and prepares it. At every step of its journey, all parties should review the radio module to identify the best key scheme to use. Let's assume that keys can be modified at three distinct layers in the following order:

1. Radio Manufacturer
2. Wireless Device Manufacturer
3. End User

Each layer has its own set of responsibilities and selections, and every selection affects the subsequent layer and final key scheme.

Radio Manufacturer

This entity manufacturers the radio module and typically provides OEM modules for end users or wireless device manufacturers.

1. **Vendor Identification (VID) Key in Firmware**: A radio manufacturer can program a unique Vendor Identification (VID) number into the radio module firmware for a fee. (For example, *Digi* offers this service for a set-up fee of $1,000 and an additional $5 per module.) Unique VIDs are assigned to only one radio manufacturer, and every radio module transmission contains a 16-bit VID that can only be configured at the factory. Radio modules can only interact with other radio modules having a matching VID--from the same wireless device manufacturer. Clearly this is a problem in that all consumers of this radio module from this manufacturer will have to share the same secret. An attacker who has purchased a radio module from the same manufacturer can, as a result, communicate these radio modules.
2. **No Encryption Key:** Having a radio module without an encryption key is a reasonable choice, because the user (or wireless device manufacturer) is responsible for setting up and changing it at will. These radio modules often come without an encryption key configured by default.

Wireless Device Manufacturer

This entity manufactures the industrial wireless devices and assembles radio modules into the devices.

1. **Per-client Encryption Key**: The wireless device manufacturer configures a unique encryption key on the radio module for each of its clients. The wireless device manufacturer then tracks this client/key relationship and makes sure that the key is preconfigured on the correct client. This is a good approach, because the end user does not need to change the key after purchase. The wireless device manufacturer can communicate with the device even after the client has deployed it on their site.

2. **Device Company Encryption Key**: Some wireless device manufacturers offer a free alternative to providing a VID key in the radio module. The wireless device manufacturer sets a default encryption key using a predefined or automated script to every device that ships from their facility. No per-client tracking is required or performed. This is a safer alternative, because as we have discussed, sharing a secret among devices is very insecure.

3. **No Encryption Key:** Having a device without an encryption key could be dangerous. If the consumer fails to configure the radio encryption settings, the device warranty may become void. To configure these settings, consumers must connect the radio module to a USB interface.

End User

This is the end user of industrial wireless devices. After configuring the devices, the end user deploys them to their site.

1. **Per-client Encryption Key**: The commissioning tool automatically configures the device's encryption parameters with a random and strong key. No user interaction is required. Alternatively, end users can use a special encryption key for each site. This is the best scenario, because the shared secret will exist between devices within the same geographical location.

2. **Device Company Encryption Key**: The end user leaves the device company encryption key configured as is on the radio module. The wireless device manufacturer maintains the end user's stored secrets (as well as those of every other end user having this device).

3. **Change the Encryption Key**: The end user voids the device's warranty when removing the radio module to change the key. Sometimes, the commissioning tool uses the Radio API to configure the encryption key with a randomly generated one.

4. **No Encryption Key**: The end user is not able to configure the radio module, and the device manufacturer did not preconfigure it. The device's commissioning tool does not set an encryption key when configuring every node.

Wireless networks do not always involve the use radio encryption keys to secure communications. Cryptography is used at higher levels of the protocols and managed by the device itself. As we have seen, radio encryption keys involve the use of shared secrets; these should only be used for the initial link. After this, other keys (such as session keys, master keys, and handheld keys) should be used.

Other encryption keys stored on the device are vulnerable to the same types of attack and issues. For example, if the key is stored on the device's main firmware, all devices sharing the same firmware share the same key and secret. This is a problem in that some attack methods reuse radio keys to attack other devices sharing the same firmware (rather than perform a hardware hack to extract the key).

Because secrets are often shared among devices, attackers follow this methodology:

1. Purchase the same device used by the target client. In doing so, the attacker will possess the same secret as the target client somewhere on his radio.
2. Carefully extract the radio module from the device without triggering the anti-tampering mechanism.
3. Connect the radio module to a USB interface. This is a trivial process, and development boards exist that can assist the attacker with this process.
4. Interact with the radio using AT commands or special APIs.
5. Send and receive frames. Using the unknown key, the radio should operate as a configured device for this purpose.

ZigBee Key Distribution

ZigBee is a wireless technology developed as an open global standard to address the unique needs of low-cost, low-power, wireless machine-to-machine (M2M) networks. The ZigBee standard operates on the IEEE 802.15.4 physical radio specification and operates in unlicensed bands, including 2.4 GHz, 900 MHz, and 868 MHz.

The Institute of Electrical and Electronics Engineers (IEEE) approved the 802.15.4 specification on which the ZiBee stack operates. This specification is a packet-based radio protocol intended for low-cost, battery-operated devices. The protocol allows devices to communicate in a variety of network topologies and can enhance battery life to several years.[6]

To operate securely in a network, a device must have a counterpart device that it can trust to obtain keys and that controls access. The ZigBee Coordinator product meets this requirement with the concept of a *Trust Center*.

The Trust Center:

- Stores the keys for the network.
- Uses security services to configure a device with its key(s).
- Uses security services to authorize a device onto the network.
- Periodically creates a new network key and broadcasts it as encrypted with the old network key

For every transaction, the originator and recipient share the same symmetric key. ZigBee uses two methods to distribute keys to originators and recipients:

- Pre-installation
- Over the air (OTA)

Installers set pre-installation keys on the device using an out-of-band method or *commissioning* tool. Commissioning involves physically deploying, addressing, and logically binding nodes to form a functional network. This covers a wide range of tasks including:

- Surveying the radio and physical environment
- Placing devices
- Configuring parameters
- Binding applications

Most pre-installation methods require installers to reflash the device in order to change the key. This is a sub-optimal procedure considering the limitation of available flash-write devices.

Installers can typically run a commissioning tool on a laptop or PDA, which is designed to make the installer's life easier by providing an intuitive user interface that hides the complexity of the underlying technology. The tool also gives installers the flexibility to visualize their network and devices and provides options to configure, commission, and manage the system.

Using the OTA method, the Trust Center sends the key to the device in plain text[7] if no other secret has been previously shared between them.

ZigBee uses three types of keys to manage security:

- **Link key**. This is used to secure unicast communication between two devices. This key protects frames at the application support sub layer, which provides a data service to the application and ZigBee device profiles. One of these devices is normally the Trust Center. The Trust Center role is established dynamically using a key establishment service.
- **Network key**. This is used to secure broadcast communication by sharing the same 128-bit network key with all network devices. If this key is pre-installed, it is moved to RAM when a device boots. An attacker can capture a node and use a hardware debugging interface tool (such as Bus Pirate or GoodFET) to extract memory data even when the chip is locked. The attacker could leverage known vulnerabilities for

this purpose such as the ones that Travis Goodspeed discovered on some microcontrollers.

OTA keys are sent encrypted on “High Security” mode. (In “Standard Security” mode, they can be transmitted as either encrypted or decrypted.) Frequently rotating keys prevents attackers from hacking them. However, the Trust Center typically sends these keys in plain text OTA. An attacker can perform an OTA attack by having a device mimic a node on the network in order to collect packets for further analysis and decryption. An effective attacker toolset is KillerBee[8] (created by Joshua Wright). This is a suite of software tools that allow 802.15.4 packets to be intercepted, analyzed, and injected with a flexible framework.

- **Master key**. This can be used as an initial shared secret between two devices when they perform a Key Establishment Procedure (SKKE) to generate link keys.

Exploiting Vendor 1 Wireless Devices

Vendor 1 wireless devices operate in high-interference environments by combining advanced frequency hopping and digital signal processing technology with outstanding receiver sensitivity and different types of antennas. This combination results in very good noise and interface rejection. Vendor 1 literature states:

"Thousands of industrial plants worldwide are operating more efficiently today with Vendor 1's integrated wireless solutions. From simple monitoring systems to high-speed control, Vendor 1's knowledge and expertise guarantees a reliable and secure industrial wireless network."

Vendor 1 industrial frequency hopping solutions include:

- Proprietary frequency hopping spread spectrum (FHSS) with 128-bit AES encryption
- Repeater mode, 1W power
- Hazardous location approvals and self-healing networks
- 30-plus mile point-to-point range with high gain antennas
- Ethernet and serial device connectivity
- Smart switches for peer-to-peer networking

Company Profile

Vendor 1 specializes in the development of communication solutions that are compatible with large automation supplier controllers. Vendor 1 provides connectivity and communication solutions that offer a seamless bridge between a variety of automation products.

Vendor 1 Tool Key Distribution

The Vendor 1 configuration tool provides a graphical representation of a radio network to make it easier to set up radios and monitor their performance. Companies can use this tool to support configuration and installation and monitor system performance on a long-term basis. Companies also commonly use Vendor 1 in long- or short-range wireless SCADA applications.

Vendor 1 supports several radio models (approximately 15 different products) manufactured by a variety of vendors. This is a statement from a Vendor 1 datasheet:

> *<Tool> is easy to use and intuitive. Default values built into the software work well for initial installation and testing making it easy for first-time users. <Tool> manages all important settings to ensure that the network performs correctly.*

When creating a new radio network, Vendor 1 generates a random passphrase and automatically sets the encryption level to 128-bit AES (see Figure 1).

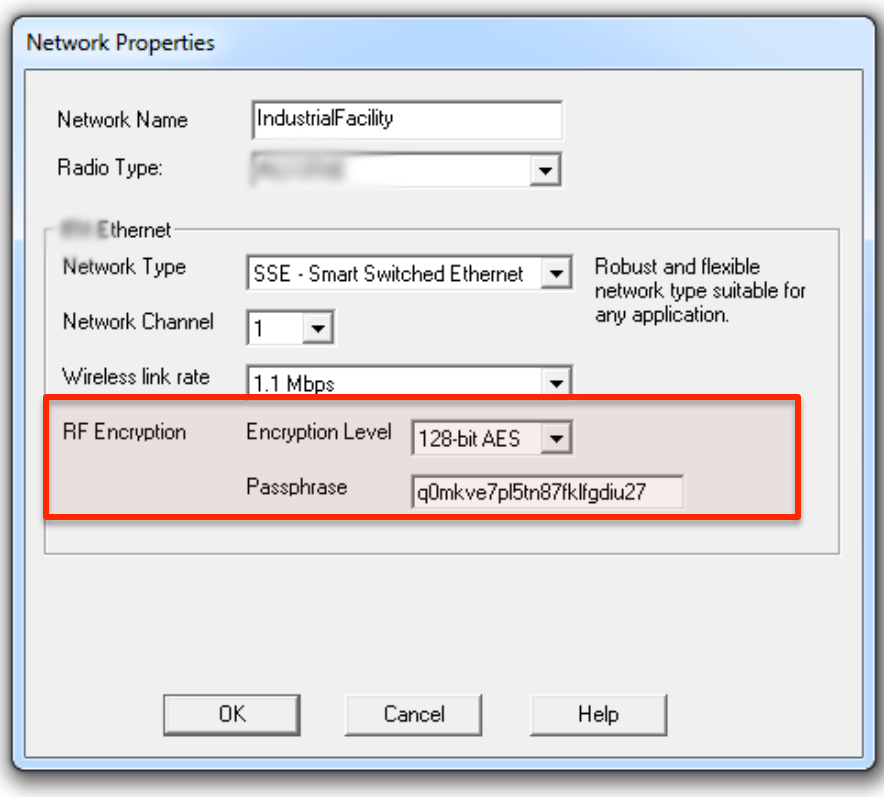


Figure 1: *New Network Wizard*

In our research, we identified a vulnerability within the underlying implementation of the pseudo-random number generator (PRNG) that generates the passphrase. A PRNG is a deterministic algorithm capable of generating sequences of numbers that approximate the properties of random numbers. Each sequence is completely determined by the initial state of the PRNG and the algorithm for changing the state. Most PRNGs make it possible to set the initial state, which is also called the *seed state*. Setting the initial state is called *seeding* the PRNG.[10]

This figure shows is how we reverse engineered the function responsible for calculating “random” passphrases (see Figure 2):

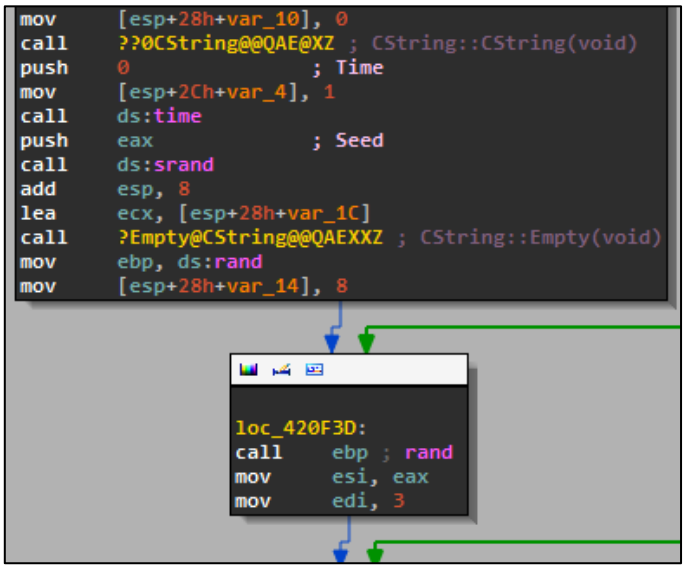


Figure 2: *Tool Binary Disassembly (v5.16.038)*

1. The PRNG is seeded using the argument passed as *seed*. The *srand* function sets the starting point for generating a series of pseudorandom integers, and subsequent calls to *rand* generate the same succession of results. To create the same sequence of results, we called the *srand* function and used the same *seed* argument again.
2. After reverse engineering the function, we developed the following C code to locally calculate a passphrase using an *epoch* value and iterate over all the possible values. This was possible, because the Windows C Run-Time Library uses the same algorithm for seeding the PRNG across almost all versions of this operating system.

```
void *printPassphrase(time_t epoch)
{
    char buff[100];
    strftime (buff, 100, "%Y-%m-%d %H:%M:%S", (int*)localtime(&epoch));
    printf("%s => ", buff);
    char passphrase[25] = "\0";
    srand(epoch);
    int block_counter = 8;

    do{
        int i = rand();
        int counter = 3;
        do{
           int i2 = i & 0x1f;
           if(i2 >= 0x0a){
               i2 = i2 + 0x57;
            }else{
               i2 = i2 + 0x30;
            }
            appendchar(passphrase,sizeof passphrase,(char) i2);
            i = i >> 5;
            counter--;
        }while(counter>0);
        block_counter--;
    }while(block_counter>0);
    printf("%d => %s\r\n", epoch, strrev(passphrase));
    return;
}
```

Figure 3: *Passphrase Generator*

3. We compiled and executed the program to obtain valid passphrases (current to past). Within a few seconds, the tool provided us with the passphrases:

```
C:\>passgen.exe
   2013-04-04 21:39:08 => 1365136748 => knc6gadr40565d3j8hbrs6o0
   2013-04-04 21:39:07 => 1365136747 => nir3f1a0dm2sdt41q91c06nt
   2013-04-04 21:39:06 => 1365136746 => qeb0dp65mc0hmd5gc1ms36np
   2013-04-04 21:39:05 => 1365136745 => t9qtcg2b01u6ut5utpcc76nm
                                  ...
   2013-04-04 20:53:46 => 1365134026 => q0mkve7pl5tn87fklfgdiu27
   2013-04-04 20:53:45 => 1365134025 => ss6hu63uurrcgng3775tlu24
   2013-04-04 20:53:44 => 1365134024 => vnlest048hp1p7hhourdpu20
```

Figure 4: *Passphrases*

4. We extracted the passphrase "*re84q92vssgd671pd2smj8ig"* from a screenshot of the oldest *Tool Help Manual we could locate*. Using the passphrase generation tool that we had developed, we identified the date the passphrase was created: *2008-04-17 15:20:47 (1208470847*). Assuming this date refers to the oldest date possible, the following is a comparison of possible attacks:

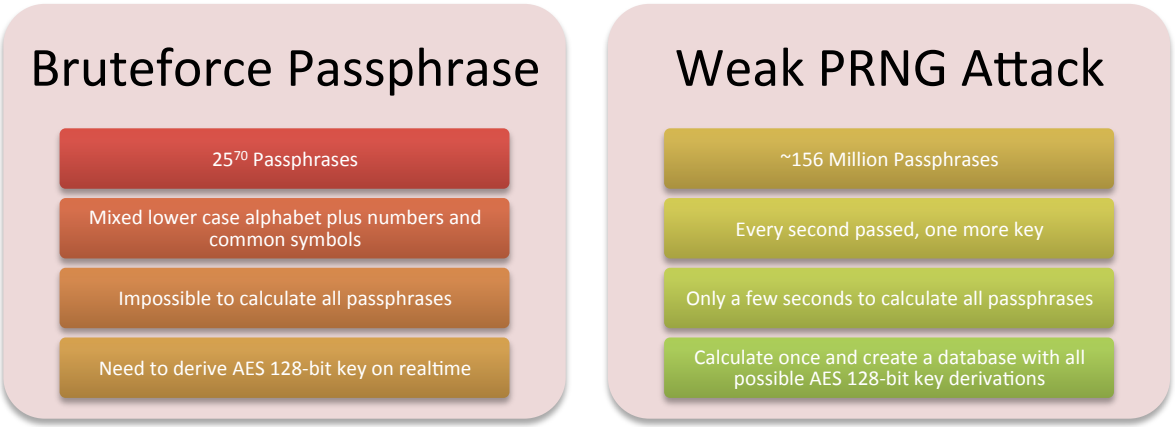


Figure 5: *Attack Comparison*

We determined that the key space was significantly reduced, which makes a weak PRNG attack feasible.

Note that custom passphrases are not vulnerable to these attacks, because the user enters these directly. However, because "*default values that are built into the software work well for initial installation*, Vendor 1's worldwide device networks use (for the most part) any of the ~156 Million passphrases or their AES 128-bit derived keys. As a result, an attacker might be able to compromise the device network and affect its data integrity, confidentiality, and availability.

Exploiting Vendor 2 Wireless Devices

Company Profile

Vendor 2 is a widely known provider of wireless automation solutions for industrial applications involving process monitoring and control systems. Their patented system includes wireless tank monitoring, wellhead monitoring, and peer-to-peer oilfield automation solutions. Forming the foundation of a highly scalable, peer-to-peer wireless infrastructure, the system enables “last mile” connectivity in any wireless SCADA and telemetry application. Moreover, devices can be deployed to implement the entire range of capabilities presented in the “Industries and Applications” section of this paper.

The Vendor 2 family of devices consists of wireless transmitters, wireless gateways, I/O expansion modules, and hardwired sensors (see Figure 6).

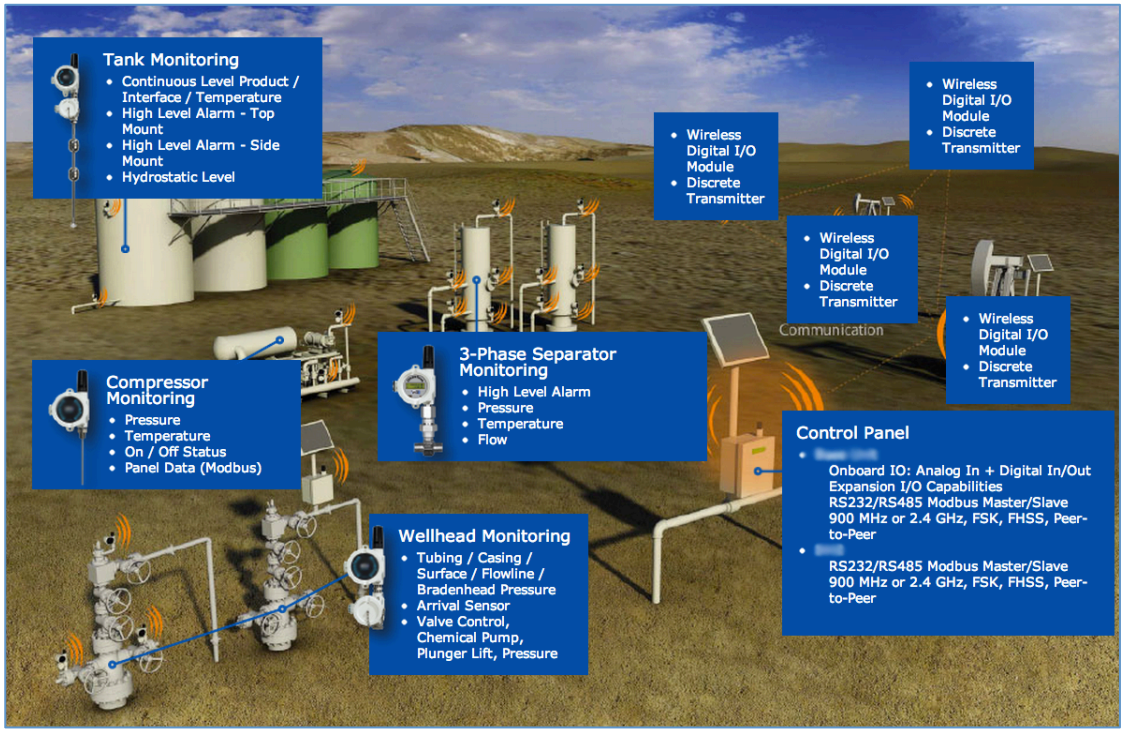


Figure 6: *Digital Oilfield Automation*

Vendor 2 transmitters are certified for “Class I; Division 1 hazardous locations”. They are battery-powered and self-contained and do not have a risk of causing an explosion. These transmitters wirelessly communicate collected data through the Modbus protocol through one of the following:

- Wireless gateway to a third-party programmable logical controller
- Remote terminal unit
- Human machine interface

- Distributed control system in a SCADA system

Devices Family Key Distribution

We researched how key distribution is implemented on a widely used set of Vendor 2 devices. Each specific field location where devices are deployed for use has a project configuration file that has been set up and saved using this software. This project file contains all the network information for the devices and is pushed to all the devices using a special cable to give them knowledge about the layout of other nodes in that location. This allows every node to communicate and export required information to other selected peers.

We located this definition in Vendor 2's glossary of terms:

> *Enhanced Site Security - Enabling site security reduces the chance that transmitted information can be **accessed by unauthorized devices** or cross-talk between other devices operating in the area. By default, site security is enabled and it is recommended to keep this default setting.*

What authorizes a device to access transmitted information? A pre-shared key might be generated within the project file, since it is also deployed to all the devices at the same site. This mechanism will also allow a variety of networks to be deployed within the same physical area without allowing cross talk between the devices.

> *The Enhanced Site Security feature is designed to provide an additional **level of protection** for **RF packets** sent and received between <Family> System devices and minimizes the possibility of interference from other devices in this area. This feature is **not available on some older versions** of legacy devices.*

Moreover, this security mechanism is not available on "legacy" devices of the same device family. From a development perspective, Vendor 2 introduced security on the family of devices in a late phase of a mature software and hardware project.

We found more evidence that Vendor 2 devices use a pre-shared key in a Vendor 2 setup guide:

> *1. If the project file name is **changed**, a new **Site Security Key** will be assigned.*
> *2. DO NOT change the file name if you are planning to use the retrieved file to update the Gateway you retrieved the file from. Other Wireless End Nodes that may be connected to that Gateway will no longer be connected if the **Security Key** has been changed.*

As indicated in one Vendor 2 guide, two field location file projects will always have a different *Site Security Key*, because the file names are different. On an NTFS file system, two files cannot have the same file name. One solution is to generate a new Site Security Key every time the name changes. This prevents sites from colliding with another project within the same physical space. With this in place, we were able to track the generation on a project file using system calls or standard libraries as a reference.

Because the portable executable file creates the project files, we began by reverse engineering the functions related to *Project File Saving* using a static and dynamic approach. We identified the basic blocks involved in the key generation process, because it uses the same logic as in Vendor 2's documentation about the Site Security Key.

This image illustrates the disassembly involved in the key generation process when a project file is saved. The code first verifies whether this feature is enabled (see Figure 7).

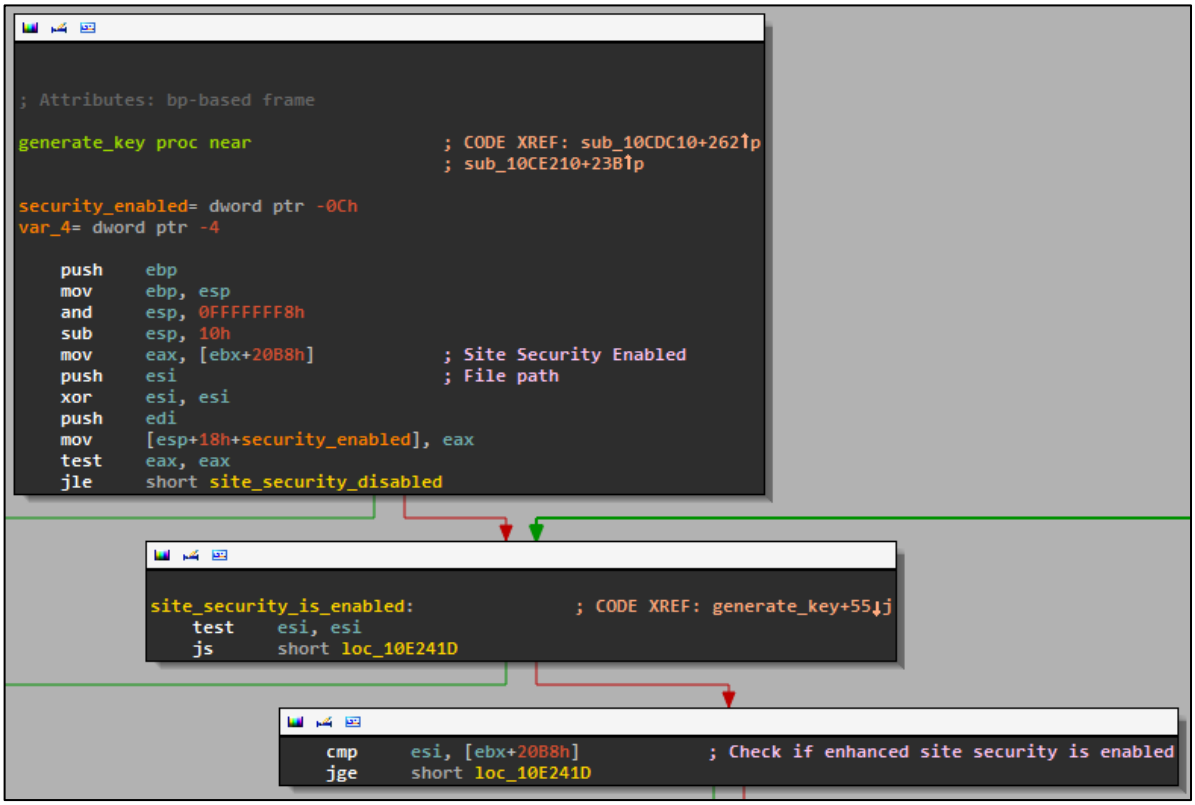


Figure 7: *Enhanced Site Security Can Be Enabled or Disabled by the Device Admin*

After this verification, the code performs a multibyte string comparison to check for the return value of a previous *mbscmp* call between the old and new project file name.

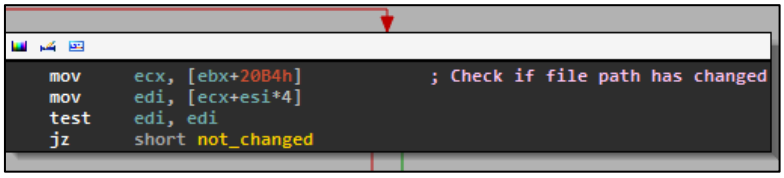


Figure 8: *File Path Change Check*

The knowledge we gained from Vendor 2's documentation helped us to determine that the routines corresponded are involved in the key generation. Moreover, the key is finally generated (see Figure 9):

```
push    0                               ; Time
call    __time64                        ; Determine the current calendar time
add     eax, esi
add     esp, 4
mov     [esp+18h+var_4], edx
mov     edx, [edi]
push    eax
mov     eax, [edx+10h]
mov     ecx, edi
call    eax                             ; eax = &update_key
mov     eax, [esp+18h+security_enabled]
```

Figure 9: *Key Generation Code*

Figure 9 above shows that the project file is directly updated with the return value of the well-known *time64* function. This function returns the value of time as seconds from January 1, 1970. Because this function is imported from the Windows library *Kernel32.dll* and calls the function *GetSystemTimeAsFileTime*, every project uses the system time as the *Site Security Key*.

We performed a binary byte-level diffing over two similar projects saved with different file names: *Project A* (see Figure 10) and *Project B* (see Figure 11). We have highlighted the *Site Security Key* in Red.

```
-- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00000000   0300 0000 0100 0000 0200 ffff 0800 0800  ................
0x00000010   4669 6c65 4e6f 6465 0000 0000 8403 0000  FileNode........
0x00000020   0000 0000 0000 0000 0000 0000 0000 0010  ................
0x00000030   0000 0850 726f 6a65 6374 4103 433a 5c01  ...ProjectA.C:\.
0x00000040   00ff ff03 0004 0053 6974 6517 584f 5101  .......Site.XOQ.
0x00000050   0653 6974 655f 3100 0000 0001 0000 0000  .Site_1.........
0x00000060   0000 0000 ffff 0800 0b00 [redacted]      ..........[redacted]
0x00000070   [redacted]  02 0000 0084 0300 0001 0000  [redacted]..........
0x00000080   0009 4761 7465 7761 795f 3100 0000 0000  ..Gateway_1.....
0x00000090   0000 0082 0010 0000 0100 0000 1600 0000  ................
0x000000a0   3c00 0000 0000 0010 0000 0000 0000 c842  <..............B
0x000000b0   0300 0000 0010 0000 0000 0000 c842 0300  ............B..
```

Figure 10: *Project A*

Figure 11: *Project B*

As we have illustrated in the following summary, both projects shared a similar key; we created both within a short time:

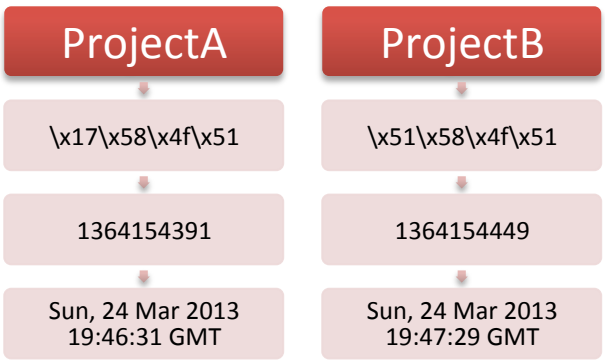


Figure 12: *Projects A and B Summary*

Leveraging the limited number of possible valid keys, an attacker could deploy a known plain text attack over the air (OTA) to identify the Site Security Key—in the event that the RF packets protection uses encryption (as specified in Vendor 2 documentation). The attacker, armed with this key, could communicate with the devices and launch other attacks over the nodes.

If the way in which Vendor 2 devices operate is aligned with Vendor 2 documentation, Vendor 2 devices could rely on a *Per-Site Encryption Key* scheme. This is the strongest mechanism we identified and analyzed on Vendor 2 devices during our research. Using this methodology, the only way an attacker can communicate with the devices is by capturing a node from the victim site and performing an expensive hardware hack to extract the encryption key.

Firmware Reverse Engineering

Vendor 2's documentation and datasheets provided us with useful information to identify critical components and subsystems. A researcher (or an attacker) can also crawl the Internet for specific information about Vendor 2, their design documents, marketing materials, patents, and support files in order to build a useful knowledge base. We obtained this information from a variety of Vendor 2 device datasheets:

HARDWARE FEATURES	
Device Functionality	· Wireless Gateway
Embedded Controller	· 32-bit Low Power ARM7 Microcontroller with Internal FLASH (Field Upgradeable)

HARDWARE FEATURES	
Device Functionality	· RTD Temperature Monitor w/ Built-In Wireless Transmitter
Embedded Controller	· Ultra-low Power RISC Microcontroller with Internal FLASH (Field Upgradeable)

Figure 13: *Vendor 2 Device Datasheet Information*

Most of the transmitters and wireless modules use an ultra-low-power Reduced Instruction Set Computer (RISC) microcontroller. This makes perfect sense, because their low power consumption capabilities will extend battery life in portable measurement applications.

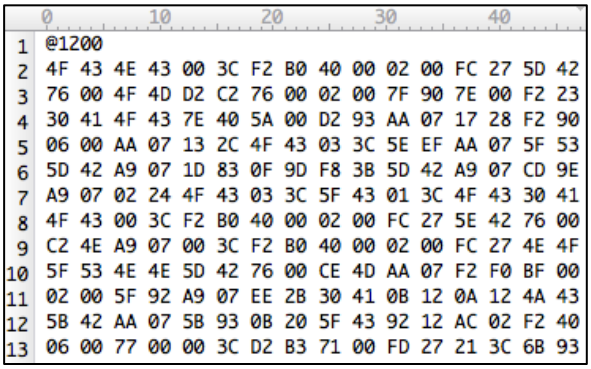
During our analysis, we identified only two types of microprocessor architectures:

- Texas Instrument MSP30
- ARM7

Steps to Identify the Microprocessor Architecture

We followed these steps to identify the microprocessor architecture:

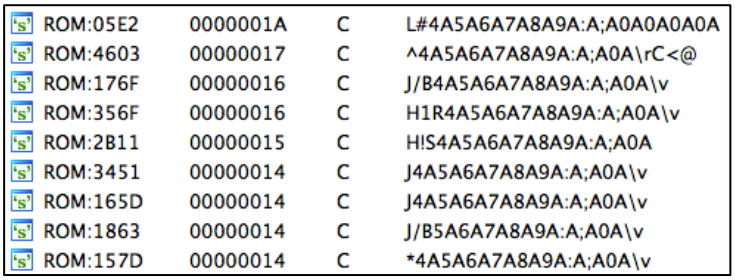
1. First, we converted the *.MEM* file to a binary file and analyzed it. The firmware dump provided by the vendor is in an industry-standard format, which consists of two basic elements: a hexadecimal address specifier and hexadecimal data values. An address specifier is indicated by an @ character followed by the hexadecimal address value. Data values can consist of as many hexadecimal characters as required. However, at least one data value must follow an address.



```
      0        10        20        30        40
1   @1200
2   4F 43 4E 43 00 3C F2 B0 40 00 02 00 FC 27 5D 42
3   76 00 4F 4D D2 C2 76 00 02 00 7F 90 7E 00 F2 23
4   30 41 4F 43 7E 40 5A 00 D2 93 AA 07 17 28 F2 90
5   06 00 AA 07 13 2C 4F 43 03 3C 5E EF AA 07 5F 53
6   5D 42 A9 07 1D 83 0F 9D F8 3B 5D 42 A9 07 CD 9E
7   A9 07 02 24 4F 43 03 3C 5F 43 01 3C 4F 43 30 41
8   4F 43 00 3C F2 B0 40 00 02 00 FC 27 5E 42 76 00
9   C2 4E A9 07 00 3C F2 B0 40 00 02 00 FC 27 4E 4F
10  5F 53 4E 4E 5D 42 76 00 CE 4D AA 07 F2 F0 BF 00
11  02 00 5F 92 A9 07 EE 2B 30 41 0B 12 0A 12 4A 43
12  5B 42 AA 07 5B 93 0B 20 5F 43 92 12 AC 02 F2 40
13  06 00 77 00 00 3C D2 B3 71 00 FD 27 21 3C 6B 93
```

Figure 14: *29-0259-001_B, RTD Transmiter_v2.2.0.8.drm*

After we converted the ASCII characters into a hexadecimal format and aligned the addresses, we found these strings within the binary:



's'	ROM:05E2	0000001A	C	L#4A5A6A7A8A9A:A;A0A0A0A0A
's'	ROM:4603	00000017	C	^4A5A6A7A8A9A:A;A0A\rC<@
's'	ROM:176F	00000016	C	J/B4A5A6A7A8A9A:A;A0A\v
's'	ROM:356F	00000016	C	H1R4A5A6A7A8A9A:A;A0A\v
's'	ROM:2B11	00000015	C	H!S4A5A6A7A8A9A:A;A0A
's'	ROM:3451	00000014	C	J4A5A6A7A8A9A:A;A0A\v
's'	ROM:165D	00000014	C	J4A5A6A7A8A9A:A;A0A\v
's'	ROM:1863	00000014	C	J/B5A6A7A8A9A:A;A0A\v
's'	ROM:157D	00000014	C	*4A5A6A7A8A9A:A;A0A\v

Figure 15: *IDA Strings Window*

In our analysis, we determined that the *"5A6A7A8A9A:A;A0A"* string was generated by the C compiler of "CrossWorks for MSP430," which is a complete development solution for MSP430 projects.

2. Next, we identified the component. It is well known that hardware vendors typically publish part numbers on the component's surface, in public documentation, and within high-resolution online images. This type of information helped us identify the architecture and correctly disassemble the firmware dumps.

   We obtained the following images from public datasheets uploaded to Vendor 2's website. By zooming in on the images, we were able to identify the microcontroller version (see Figures 16 and 17).

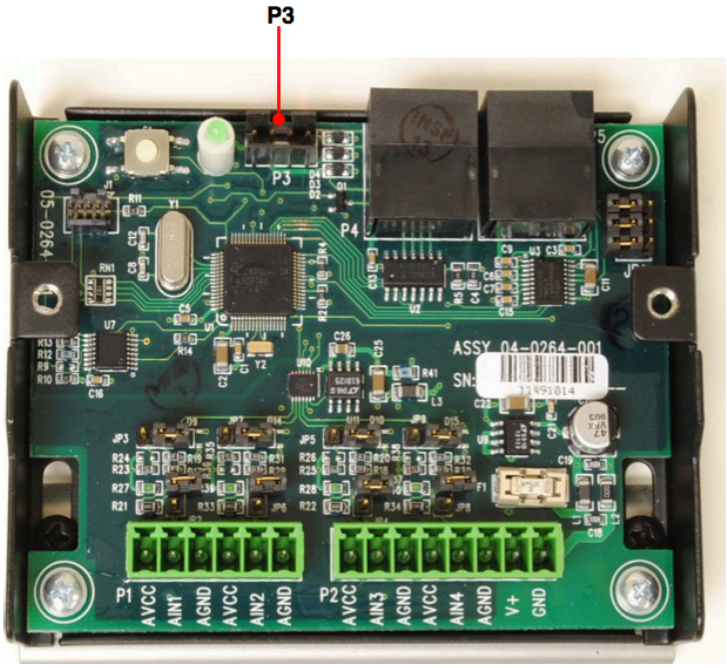


Figure 16: *Image of one module captured from a User Guide*

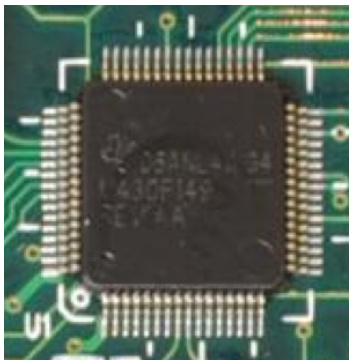


Figure 17: *430F149 (CPU)*

Texas Instrument MSP30

The first architecture we identified was the Texas Instrument MSP430 CPU, which is a 16-bit RISC architecture that is highly transparent to the application. All operations (with the exception of program-flow instructions) are performed as register operations in conjunction with seven addressing modes for the source operand and four addressing modes for the destination operand.

The CPU is integrated with 16 registers that provide reduced instruction execution times. The register-to-register operation execution time is one cycle of the CPU clock. Four of the registers (R0 to R3) are respectively dedicated as the program counter, stack pointer, status register, and constant generator. The remaining registers are general-purpose registers.

Typical applications include sensor systems that capture analog signals, convert them to digital values, and process and transmit the data to a host system.

ARM7

The second architecture we observed was a type of wireless gateway firmware that uses the *Intel Hex File Format*. This file format is one of the oldest available but is still adopted by newcomers to the market. The Intel Hex File Format was originally designed for a 16-bit address range (64kb). Later, the file format was enhanced to accommodate larger files with 20-bit address ranges (1M) and even 32-bit address ranges (4G).

All data lines are called records, and each record contains these fields:

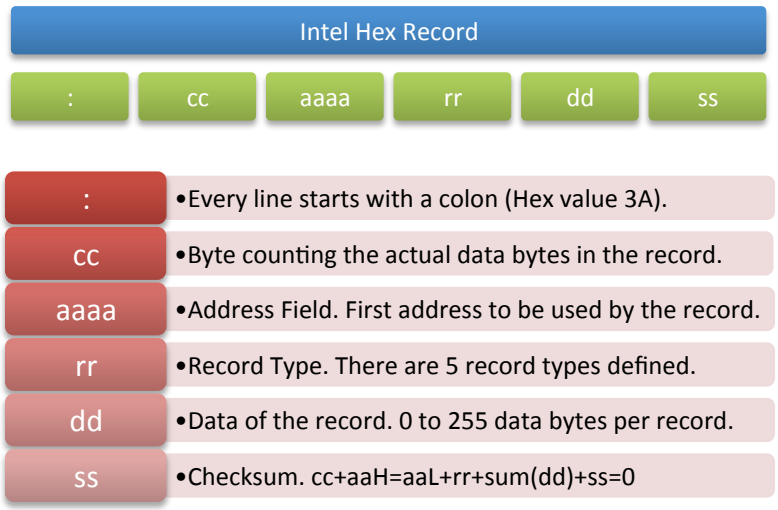


Figure 17: *Intel Hex Record Fields*

We used these steps to extract the true firmware image:

1. Identify “data records” by searching for records with record type “00”.
2. Extract the data from those records.
3. Align the data using the address field, and pad all non-contiguous memory with “0”.
4. Dump the output to a file.

We loaded the firmware image in IDA and specified the ARM processor module to correctly read the file’s header to organize memory. IDA recognizes the following:

- Areas of memory code or data
- Blocks of code as functions
- Library code as functions (using pattern-matching techniques)

The *navigation band* is an IDA component that presents a linear view of the address space of the loaded file. Different colors[2] represent different types of file content, such as data (grey), code (light blue), red (instruction), or unexplored (green).

In our analysis, IDA failed to define most of the firmware image functions (see Figure 18)..

Figure 18: *Navigation Band Prior to Executing the IDA Script*

Because IDA integrates scripting features, the process of accessing its database using different languages is straightforward. One can use the following steps to apply a well-known algorithm to any processor architecture to identify ARM function prologues not recognized within the image:

1. Locate the *opcodes* prologue for already defined functions.
2. Remove incorrect prologues (incorrectly defined functions may exist).
3. Search the entire binary for such function prologues.
4. Mark matches as code and create functions using the *MakeFunction* command.

The following navigation band image illustrates the result of this algorithm:

Figure 19: *Navigation Band after Executing the IDA Script*

The algorithm revealed that many additional defined functions existed. As a result, we were able to use the IDA cross-referencing features on the binary image.

Searching Cryptographic Keys

At this point in our analysis, we had two types of device firmware in our possession for different process architectures. We wanted to determine whether an embedded encryption key was stored on both images. Although the assembly code of these firmware types was very different, we were still able to use the following algorithm to search for embedded cryptographic keys:

1. Set a predefined chunk size (for example, 4 bytes).
2. Extract all possible contiguous chunks from Binary Image #1. (Increase the offset by 1 to generate a new unique chunk.)
3. Search every chunk in Binary Image #2. Convert the endian byte order (if required) before matching.

4. Repeat these steps but increase the chunk size by one each time until reaching the predefined limit (for example, 512 bytes).

5. Obtain all hits (offsets and values), and audit both disassembly codes to verify the presence of a cryptographic key.

We identified this sequence of bytes on every firmware of the device family. However, we determined that this was only a frame check sequence (FCS) lookup table and not an encryption key (see Figures 20 and 21).

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0000	C1C0	81C1	4001	01C3	C003	8002	41C2	01C6	C006	8007	41C7	0005	C1C5	81C4	4004
1	01CC	C00C	800D	41CD	000F	C1CF	81CE	400E	000A	C1CA	81CB	400B	01C9	C009	8008	41C8
2	01D8	C018	8019	41D9	001B	C1DB	81DA	401A	001E	C1DE	81DF	401F	01DD	C01D	801C	41DC
3	0014	C1D4	81D5	4015	01D7	C017	8016	41D6	01D2	C012	8013	41D3	0011	C1D1	81D0	4010
4	01F0	C030	8031	41F1	0033	C1F3	81F2	4032	0036	C1F6	81F7	4037	01F5	C035	8034	41F4
5	003C	C1FC	81FD	403D	01FF	C03F	803E	41FE	01FA	C03A	803B	41FB	0039	C1F9	81F8	4038
6	0028	C1E8	81E9	4029	01EB	C02B	802A	41EA	01EE	C02E	802F	41EF	002D	C1ED	81EC	402C
7	01E4	C024	8025	41E5	0027	C1E7	81E6	4026	0022	C1E2	81E3	4023	01E1	C021	8020	41E0
8	01A0	C060	8061	41A1	0063	C1A3	81A2	4062	0066	C1A6	81A7	4067	01A5	C065	8064	41A4
9	006C	C1AC	81AD	406D	01AF	C06F	806E	41AE	01AA	C06A	806B	41AB	0069	C1A9	81A8	4068
A	0078	C1B8	81B9	4079	01BB	C07B	807A	41BA	01BE	C07E	807F	41BF	007D	C1BD	81BC	407C
B	01B4	C074	8075	41B5	0077	C1B7	81B6	4076	0072	C1B2	81B3	4073	01B1	C071	8070	41B0
C	0050	C190	8191	4051	0193	C053	8052	4192	0196	C056	8057	4197	0055	C195	8194	4054
D	019C	C05C	805D	419D	005F	C19F	819E	405E	005A	C19A	819B	405B	0199	C059	8058	4198
E	0188	C048	8049	4189	004B	C18B	818A	404A	004E	C18E	818F	404F	018D	C04D	804C	418C
F	0044	C184	8185	4045	0187	C047	8046	4186	0182	C042	8043	4183	0041	C181	8180	4040

Figure 20: *Lookup Table*

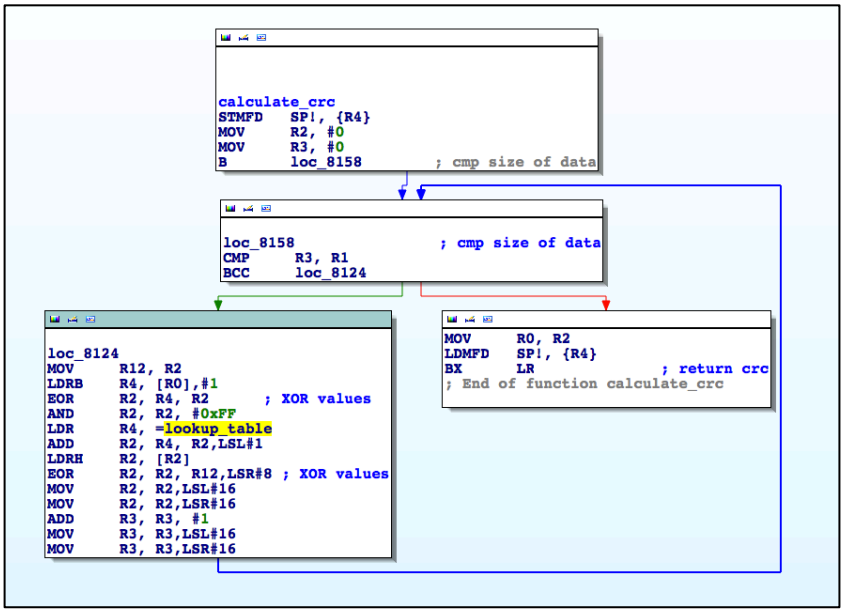


Figure 21: *ARM Function Disassembly (Firmware)*

We determined that no other relevant byte sequences were shared among the device family and (possibly) no cryptographic key existed on the firmware images.

Acquiring the Devices

Our decision to acquire a wireless gateway and RTD temperature transmitter was based on a number of hypotheses, theories, and suppositions; in our research, we discovered that we were correct about many of these. Vendor documentation, software, and firmware images were available to us throughout this project.

An attacker could easily guess the level of encryption being used on the devices or radio modules before acquiring them by identifying the hardware components. Anyone can download commissioning tools for free and audit these to determine the key distribution schemes in use.

As we discussed in the section “The Journey of Radio Encryption Keys”, if no apparent key is saved on each node during commissioning, the vendor might instead rely on a Radio *Encryption Key* or *Firmware Stored Key*. Unless the vendor has already flashed a special firmware for the end user, such keys are typically weak. This is because they are shared among multiple end users and companies.

Before acquiring the actual devices for our analysis, we assumed that this shared secret was actually the *Site Security Key*. This would the perfect encryption-based scheme candidate, because the commissioning tool dynamically generates and distributes this key over a secure channel (through a *wired debug port*). We were motivated by the possibility of breaking the weak key remotely over the air, as this was the only real secret shared among devices within the same location (or site).

After identifying the radio module and baseline protocol (802.15.4), we would use this Site Security Key to encrypt the data payload. Moreover, we wondered if the vendor had set an additional *Device Company Encryption Key* on the radio module after purchasing it from Digi (the radio's vendor). We were fairly certain that the vendor device did not include a *Per-Client Encryption Key*. To verify this, we contacted the vendor saying that we were going to borrow a device from someone with the assumption that it would communicate with our devices after upgrading the Site Security Key. This worked and the answer revealed to us that the vendor was using either a Device Company Encryption Key or no encryption at all. The following is an email message we sent to a device vendor reseller along with his response:

```
Dear <Reseller>,
We are going to borrow a used "Analog Transmitter" from one of our
partners and test it for a few weeks. We will let you know if we
decide to buy a new one. Are there any specific concerns we might
take into account when deploying this device to connect it with
our gateway? Or should we just upgrade all project configuration
files? Thank you.
```

```
Lucas,
You just need to upgrade the configuration files.
Thanks.
```

Because the devices are not tracked on a per-client basis, they are not pre-configured or reflashed. No other encryption key or secret is stored between the devices and vendor.

The devices we ordered arrived after several weeks. All of the hardware components we had previously identified matched perfectly with the actual components. More importantly, we had tested our theories with a higher success probability than expected.

To sniff radio packets, we used a USB transceiver and a logic analyzer. Because this transceiver did not support packet injection by default, we reprogrammed the microcontroller with a modified firmware from the same vendor using a development tool that supports JTAG.

Authentication Bypass in Family of Devices

We developed a custom tool to send and receive 802.15.4 data. For this purpose, we used some of the following python modules:

- Scapy dissectors
- KillBee Frame Check Sequence code
- PyUsb
- IPython (for the user interface)

We expected to find a per-site encryption key but realized no encryption was performed on the transmitted RF data. Using our python tool, we intercepted the Site Security Key, which is prepended in some packets and used to authenticate the devices.

This sequence of bytes allowed us to impersonate more devices of the family by using this value with:

- The corresponding assigned ID
- A specific device opcode (found on the device firmware images)
- A specific group of values (varying by sensor type)

This authentication bypass vulnerability allowed us to inject any frame into the gateway nodes and overwrite the internal Modbus register table. We did so by reverse engineering the firmware to understand how those values were mapped from the data packet aided by debugging symbols and strings.

Forged frames injected into the gateway nodes allowed us to modify the actual readings of the values transmitted by these sensors in real time. We have provided a list of some of the transmitters that send information to gateway nodes:

- RTD Temperature Transmitter
- Pressure Transmitter
- Thermocouple Transmitter
- Liquid Level Transmitter
- Magnetostrictive Transmitter
- Flow Totalizer / Hydrostatic Transmitter

The maximum range in which this attack could success is depicted here:

Figure 22: *Maximum Range of the Device Antennas*

Racing the Modbus Master Device

When exploiting the authentication bypass vulnerability, it is crucial to have a specific window of time when the victim device receives the forged packet. This type of attack

depends primarily on the speed of the polling Master device to request a specific Modbus register.

By default, Human Machine Interfaces (HMIs) will scan the communication worksheet of the project every X milliseconds. When the forged measurement is injected on the gateway node (Modbus Slave Device), it overwrites the Modbus register table on the device for a period of time. If this new measurement value is injected after the original value and prior to the Master device polls the register, the attack will succeed. The polled value is injected instead of the original value. The probability of injecting the packet before the Master device reads the register is illustrated in this formula:

$$P(X) = 1 - \frac{i}{r \times n}$$

- *r: Master read interval*
- *i: Attacker injection interval*
- *n: Number of injection instances*

This formula illustrates that by using more injection instances, the probability of success is higher. These instances must transmit the packets at different offsets. We have provided some examples of this attack failing and succeeding. In the first image, the transmitter sleep interval has won the race, and the master read interval has used its value saved on the gateway Modbus register (see Figure 23).

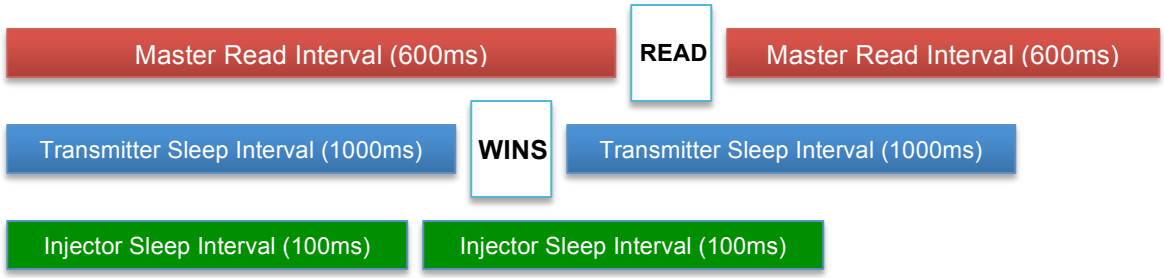


Figure 23: *Attack Failure*

The next image illustrates the injection instance winning the race (see Figure 24). The transmitter sent its message just before the injection occurred. To reduce the injector sleep interval (when it is not possible to reduce it due to the physical limitation of the device), one might consider using another instance. This will increase the probability of success, as we illustrated in the formula above.

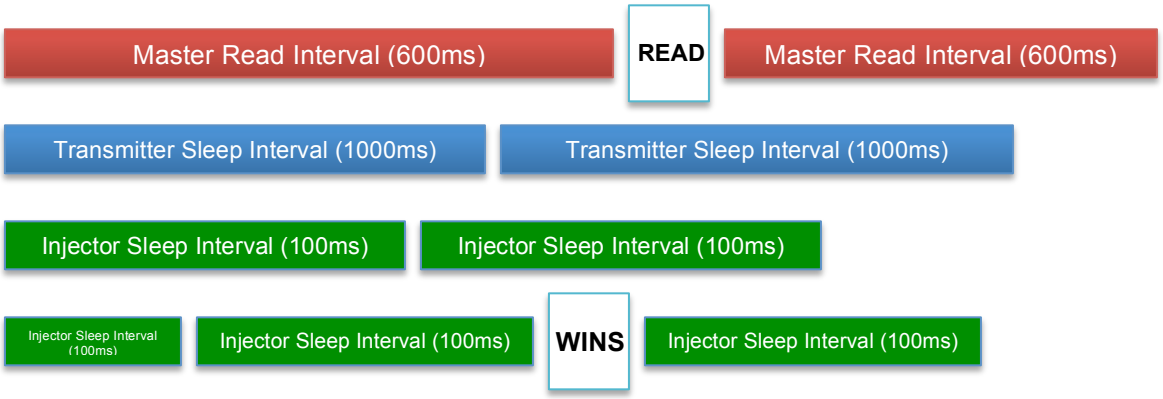


Figure 24: *Attack Success*

Memory Corruption in Wireless Gateways

A memory corruption vulnerability exists in wireless gateways that involves a protocol parsing function. Attackers can leverage this vulnerability to send a specially crafted packet over the air and disable the reception of further messages on Gateway nodes. Attackers could exploit this vulnerability to:

- Overwrite memory
- Modify existing project files
- Retrieve the device firmware

If this occurs, the only way to “unbrick” the gateway nodes is to manually reboot them; this is approximately a two-minute process per device.

An outsider can disable an entire plant or facility by exploiting this vulnerability in a persistent way to interrupt every communication between Gateways and Transmitter nodes for an undetermined period of time.

We successfully tested the attack launching the exploit over the air.

Exploiting Vendor 3 Wireless Devices

Vendor 3 offers an industrial wireless I/O network that can operate in extreme environments and eliminate the need for costly wiring. The most basic sensor network includes a *gateway system controller* and one or more nodes that monitor and/or control I/Os in remote locations.

Nodes are slave devices within Vendor 3's wireless sensor networks. Sensors and other devices connect to the node's inputs or outputs. The node collects the data and wirelessly transmits it to the gateway.

A *wireless gateway* is the wireless network master device used to control network timing, schedule communication traffic, and hold the configuration for the entire I/O sensor network. The wireless gateway acts as a portal between the wireless network and the central control process—similar to how a gateway device on a wired network acts as a "portal" between networks.

Company Profile

For 20 consecutive years, Vendor 3 placed first in more than 50 independent studies of engineer purchasing preferences. With more than 22,000 different products across 40 industries, Vendor 3 offers a complete line of:

- Photo eyes
- Wireless sensors
- Vision sensors
- Vision lighting
- Machine safety
- Indicator lights

Consumers most commonly use Vendor 3's wireless devices in the following scenarios:

- *Tank level management*. Accurately measuring tank levels, pressure, or flow rates with a device node and an external sensor.
- *Temperature and humidity control*. Maintaining environmental conditions using a node temperature and humidity sensor.
- *Compost window temperature monitoring*. Determining the optimum time to turn the windows for quicker compost production using accurate temperature measurements and data logging.

- *Wellhead pressure monitoring*. Monitoring the pressure at the wellhead by connecting pressure transducers to a device node with an integrated batter.
- *Pipeline flow measurement*. Measuring the total flow by wirelessly transmitting the rate at the source back to the office to compare it with the gas flow rate at the destination.
- *Flare stack temperature alarm*. Detecting pressure or vacuum within the methane production system using a pressure transducer. For this purpose, thermocouples are connected to nodes to detect the heat of an active flame and verify the combustion of methane.
- *Wastewater analysis*. Monitoring multiple data points including fill level, pH, conductivity, temperature, or flow using a single node with analog inputs.
- *Failed conduit replacement*. Using point-to-point radio devices. These are simple wire replacement radios, which are easy to install and do not require special programming.
- *Water tower level and alarm*. Making sure that water tower levels are constantly being monitored using an ultrasonic sensor connected to a node. When the water levels fall, pumps move more water from the reservoir to the tower.
- *Predictive maintenance*. Using mounted nodes equipped with thermocouples or RTDs near motors. Alarms alert maintenance if predetermined temperatures are exceeded.

Wireless Devices Security

Vendor 3 provides a set of catalogs and datasheets on their website about their products. We read these documents to better understand their security schemes.

We found the following information in a security guide:

> *<Vendor 3> wireless systems were designed from the ground up with Network Security, Data Security and Data Integrity at the forefront. These wireless I/O systems provide a level of security, data integrity, and reliability far exceeding most wireless systems on the market today.*

Network Security

We found the following information relating to network security within Vendor 3's security guide for the device:

> *<Vendor 3> wireless systems are designed to completely eliminate all Internet Protocol (IP) based security threats.*

> *(…) Malicious TCP/IP packets and programs can cause grave security breaches and could cause the loss or theft of critical information. This is because* ***standards-based network components such as Wi-Fi access points have the potential to route any and all data packets****, which is why these systems use* ***encryption****, passwords, firewalls, and antivirus software. Vendor's <Family> systems, however, do not pose a security threat to the network because the <Family> system cannot physically route malicious TCP/IP packets. (…)* ***The <Family> protocol only carries sensor data values. Only I/O data is transmitted in the wireless layer.***

While it is true that these devices do not route "malicious" TCP/IP packets into the internal network, the ability to route packets to the internal network is not *the reason* that Wi-Fi access points systems use encryption. Encryption provides confidentially and makes sure that information is not disclosed to unauthorized individuals, processes, or devices. Moreover, the protocol of these devices carries sensor data values—this is the most important type of information to secure but is despised in the text above.

Communication Protocols

We found the following information relating to communication protocols in a security guide:

> *Using a proprietary protocol provides a high level of security. Data security is far more of a concern when using open protocols. With an open protocol and no security encryption, anyone using that protocol can intercept and monitor your data.* ***Widely used open protocols such as Wi-Fi have serious security issues. Even a high degree of encryption may not protect your data. It is common for new encryption schemes to be hacked within months of implementation. Proprietary systems are more difficult to hack than an open standard****. <Vendor 3> developed a communication method that gives <Family> the ability to carry only I/O data signals between Nodes and Gateways.*

We believe that even with a high degree of encryption in place, Vendor 3's data may not be sufficiently protected. Attackers commonly hack new encryption schemes within a matter of months after their implementation.

Vendor 3's marketing strategy relating to communication protocols relies on the basis that open protocols are easier to understand. They do not address the ease with which these protocols can be broken. An attacker can easily carry out reverse engineering over this proprietary protocol. In fact, we believe that this proprietary protocol is much more vulnerable than open protocols.

Data Security

We found the following information relating to data security:

> *<Vendor 3> achieves data security by using a **proprietary protocol**, **pseudo-random frequency hopping**, and **generic data transfer**. The <Family> protocol only carries I/O data, making it impossible for a malicious executable file to be transmitted.*

> ***This protocol does not operate like an open protocol such as Wi-Fi and is not subject to the risks of an open protocol**. <Family> also **uses pseudo-random frequency hopping and generic data transfer without context to ensure signal integrity**.*

> *The second level of data security protection is the **pseudo-random frequency hopping table**. Each time a message is sent a new frequency is chosen, which makes it almost impossible for any system listening at a given frequency to hear more than a few messages out of hundreds.*

The level of data security discussed above is known to be an obstacle to eavesdropping or sniffing. However, many researches have proven that the *frequency hopping spread spectrum (FHSS)* is highly insecure and should not be considered a security feature[15].

> *Finally, and most importantly, the <Family> wireless system uses a concept of generic data transfer without context. **Even if a hacker managed to crack the data packet format, all they would see is a set of 16-bit numbers with no reference as to what the numbers meant. No information describing the network layout or what the sensors are monitoring is ever sent wirelessly**. A hacker, if they managed to receive <Vendor> Wireless data, would only **see the actual sensor data**, not what the sensor was reading or what role the sensor played within the wireless I/O network.*

Again, we believe the information contained in Vendor 3 packets is underestimated, because they contain only *sensor data*. Clearly, an attacker does not need to know the context of the sensor to cause trouble. An attacker could, for example, easily forge fake packets using Vendor 3's proprietary protocol to perform reverse engineering by identifying the same sequence of values after subsequent readings.

Data Integrity and Control Reliability

We found the following information relating to data integrity and control reliability:

> *To guarantee the highest possible levels of **data integrity**, the <Family> wireless system employs **binding**, **cyclic redundancy check** (CRC), **link health monitoring**, and **a preset default output state**.*

> *Binding the radios to each other adds an **additional layer of security to an already secure platform**. Binding locks radios to a specific master radio by **teaching the radios the master radio's access code**. After devices are bound, the radios only accept data from that master radio and the master radio only accepts data from those specific radios bound to it.*

We also extracted this description of Vendor 3's radio binding:

> *Binding Nodes to their Gateway ensures the Nodes **only exchange data with that Gateway**. A Gateway and its Nodes will not communicate until the Gateway teaches the Nodes the **binding code**. Verify the radios are at least three meters apart before binding the radios. Bind the radios before installing them to their final locations. On the gateway, triple click button to enter binding mode. The red LEDs flash. Any Node entering binding mode will bind to this Gateway.*

During automatic binding, Vendor 3's gateway broadcasts the binding code to all nodes currently in binding mode. This code is commissioned over the air without encryption, because both devices have not yet shared a secret.

> *When the data is transmitted, **a CRC algorithm ensures that the data arrives intact**. If the CRC algorithm fails, the corrupt data packet is discarded and the data is automatically retransmitted using a new frequency during the next communication cycle.*

We feel that CRCs are not suitable for protecting against intentional alteration of data. They are specifically designed to protect against common types of communication channel errors. In this case, the data integrity property is not available.

> *<Vendor 3> wireless system continuously monitors the health of all wireless links in the system. A "link" is defined as the real-time connection between two radios. If any link is lost, the inputs and outputs associated with the radio are set to a predefined value. **When a radio drops out of the network, from a lightning strike for example, the master radio detects the link has been lost and reports the loss to the control or monitoring application**. At the same time, the master radio sets the inputs and outputs of the radio to predefined data values, resulting in predictable network behavior during a communications error.*

If an attacker successfully sends “beacon” frames to the gateway without interruption so that the gateway never detects the link has been lost, this could prevent the *preset default output state* discussed above.

Encryption Use

Vendor 3’s documentation suggests that only one device (from their family of devices) uses encryption—the *Ethernet radio* device version. We determined this based on their product user guide in which they claim to be using 256-bit AES encryption for Ethernet data packets and a network key shared between radios using a web-based configuration page. This is consistent with their claim that they use encryption only to protect against attacks to the internal network where TCP/IP packets are routed. (See the quotes in the “Network Security” section.)

Conclusions

Our investigation revealed many interesting facts about the security of Vendor 3’s devices. Because of their lack of security, the integrity and confidentiality provided by these devices is prone to attack. Assuming that Vendor 3’s documentation on the actual workings of these devices is accurate, an attacker could easily inject fake senor data into the gateway.

If Vendor 3 were to implement a proper key distribution scheme, nodes and gateways could securely communicate without sending commissioning code in plain text the first time communication is established (before the trusted relationship is established).

We would like to note that ideally radio transceivers should have a particular key stored on them, which is shared between all devices within the device family. However, even so, an attacker could still manage to extract the radio module, reuse the key with a USB interface, and perform a frame injection.

Because Vendor 3 devices can be purchased separately and offer no device-client tracking system, a “per-client” encryption key is most definitely not available in this scheme.

Conclusions

In this paper, we have reviewed a variety of widely used industrial wireless devices in terms of security, with a primary focus on key distribution. All of these device vendors shared many of the same characteristics:

- A vague concern about security at all phases of development and design of their product family
- Faulty implementation leading to the potential of a compromised network by an outsider
- Vulnerabilities due to contradictions in their documentation relating to security features.

Most current wireless sensor networks have standards and guidelines that vendors must follow when designing key schemes. We believe that these are improving over time as they begin to require an increased number of cryptographic keys--each having a specific purpose. This variety of keys increases the complexity of a device and makes it less prone to attacks.

We discussed in this paper that it is not a trivial task to decide when and where to generate and distribute cryptographic keys. And although the device hardware is becoming more secure and fewer keys are being leaked to attackers, in some cases, shared secrets are still being used. The less often this occurs, the better.

We revealed in this paper that one vendor has implemented a very good scheme using a per-site encryption key. However, the vendor is not using these keys to protect RF packets. Other vendor, has entrusted their key generation process to their technical requirements. As a result, an attacker could easily pre-calculate all the possible encryption keys (discussed in the vendor's published documentation).

One vendor that we analyzed in this paper uses a marketing tactic to highlight specific security features exposing unfavorable opinions over open designs.

It is our conclusion that these devices may pose security threats for some time to come. However, researchers are continuing to write academic papers about the future of wireless sensor networks and key distribution. In the meantime, industrial security certification and evaluation programs should, at minimum, discuss this topic in more detail. Also, industrial companies that rely on wireless technologies should re-examine their key design and implementation processes and become aware of how secrets are stored on and shared by their devices. Companies can use, for example, transceivers and open source software projects (such as *HackRF*, *RFCat*, *Api-Do*, or *KillerBee*) to perform a basic security assessment of their protocol and communication mechanisms to uncover potential vulnerabilities. ICS-CERT (and similar organizations) should be contacted to coordinate the disclosure and remediation with the vendors, as like in all the vulnerabilities identified in this research.

Acknowledgements

1. IOActive, Inc.
2. All the authors and researchers mentioned at *References* section
3. ICS-CERT

References

1. Piotr Szczechowiak, “Cryptographic Key Distribution in Wireless Sensor Networks using Linear Pairings,” Faculty of Engineering & Computing, Dublin City University, 2010.
2. Chris Eagle, “The Ida Pro Book: The Unofficial Guide to the World’s Most Popular Disassembler,” No Starch Press, 2011.
3. Haowen Chan, “On the Distribution and Revocation of Cryptographic Keys in Sensor Networks” in IEEE Transactions on Dependable and Secure Computing, pages 233 - 247, July-Sept, 2005.
4. H. Chan, A. Perrig, and D. Song, “Key Distribution Techniques for Sensor Networks” in Wireless Sensor Networks, T. Znati, K. M. Sivalingam, and C. S. Raghvendra, Eds. Kluwer Academic Publishers, 2004.
5. Seyit A. Camtepe, Bulent Yener “Key Distribution Mechanisms for Wireless Sensor Networks: a Survey,” Rensselaer Polytechnic Institute, 2007.
6. Digi International, “ZigBee Wireless Standard: Low-cost, Low-power, Wireless Networking for Device Monitoring and Control.”
7. Elpiniki Tsakalaki, Carol J. Lallier, “Ensure Your Random Number Generator is Properly Seeded,” CERT Secure Coding, 2012.
8. Joshua Wright, “KillerBee: Practical ZigBee Exploitation Framework”, 2011
9. Rob Havelt, “Yes It is Too WiFi, and No Its Not Inherently Secure”, 2009
10. Travis Goodspeed, “Nifty Tricks and Sage Advice for Shellcode on Embedded Systems”, 2013