# Weaponizing Wireless Networks: An Attack Tool for Launching Attacks against Sensor Networks

Thanassis Giannetsos
Athens Information Tech.
19.5 km Markopoulo Ave.
Athens, Greece
agia@ait.edu.gr

Tassos Dimitriou
Athens Information Tech.
19.5 km Markopoulo Ave.
Athens, Greece
tdim@ait.edu.gr

Neeli R. Prasad
Aalborg Un.
Fr. Bajers Vej 7A5
DK-9220,Denmark
np@es.aau.dk

## ABSTRACT

The pervasive interconnection of autonomous sensor devices has given birth to a broad class of exciting new applications. At the same time, however, the unattended nature and the limited resources of sensor nodes have created an equal number of vulnerabilities that attackers can exploit in order to gain access in the network and the information transferred within. While much work has been done on trying to defend these networks, little has been done on suggesting sophisticated tools for proving how vulnerable sensor networks are. This work demonstrates a tool that allows both passive monitoring of transactional data in sensor networks, such as message rate, mote frequency, message routing, etc., but also discharge of various attacks against them. To the best of our knowledge, this is the first instance of an attack tool that can be used by an adversary to penetrate the confidentiality and functionality of a sensor network. Results show that our tool can be flexibly applied to different sensor network operating systems and protocol stacks giving an adversary privileges to which she is not entitled to. We hope that our tool will be used proactively, to study the weaknesses of new security protocols, and, hopefully, to enhance the level of security provided by these solutions even further.

## Categories and Subject Descriptors

[**Audit and Attack**]

## Keywords

Wireless Sensor Networks, Passive Inspection, Architecture Layout, Network Attack Visualization, Arbitrary Code Size Injection Attack, Routing Attacks, Program Dissemination

## 1. BACKGROUND AND MOTIVATION

During recent years, wireless sensor networks have found several applications, ranging from military to civilian and commercial ones and it is expected that their adoption will spread even more in the future. They are destined to play an important role in monitoring people, objects, and infrastructure for purposes like environmental assessment [1, 2], in-home patient care monitoring [3], and so on. What makes sensor networks attractive is that they can operate unattended and without the help of any infrastructure or interaction with a human. Also, their wireless networking nature presents many advantages because of increased accessibility to information resources.

However, it is exactly this wireless technology and unattended nature of sensor networks that creates new threats and increases their information security risk profile. Their inadequate physical protection makes them receptive to being captured, compromised and hijacked [4]. Thus, any cryptographic material they contain can be used by adversaries to perform attacks from within the network compromising data confidentiality. Moreover, since communication takes place "through the air" using radio frequencies, a wide class of attacks are enabled ranging from passive eavesdropping to active interfering [5].

Although several defense mechanisms [6, 7] have been proposed in the literature over the last few years, little work has been done to demonstrate how vulnerable, in terms of data confidentiality and network availability, these networks are. Motivated by this unexplored security aspect, we demonstrate an attack tool that can be useful not only in highlighting the importance of defending sensor network applications against attacks but also in studying the effects of these attacks on the sensor network itself. This in turn can lead to the development of more secure applications and better detection/prevention mechanisms.

Our tool allows both *inspection* of a sensor network's functionality by analyzing overheard radio messages as well as *discharge* of various attacks against it. It can identify common applied protocols and use this information for performing attacks such as *Sinkhole attack* [10], *Replay attack* [11], or *Injecting malicious code* [12, 13] in order to take control over the network. Also, it can extract useful network information such as node crashes, reboots, routing problems, network partitions, and traffic analysis (overall network traffic or overheard traffic by each sensor node).

The intuition behind the presented work is to build a tool that can be used, eventually, to compromise the *functional-*

*ity* and *confidentiality* [14] of such a network. By functionality, we mean the correct operation of all network nodes as implied by the underlying application, routing and physical layer. The described tool can disrupt this normal operation by launching a number of attacks, as mentioned previously (for more details see Section 4). Confidentiality is defined as the assurance that information is accessible only to those authorized to have access. Our tool threatens the privacy of transactional data since it gives an adversary the ability to learn information by the mere presence of a message being transmitted. The data gathered from these networks can be analyzed to extract important information regarding objects, events, and individuals.

Overall, the objective of this work is to show how vulnerable sensor networks are against sophisticated attack tools and emphasize the need for appropriate prevention and/or detection mechanisms.

## Our Contribution

This work tries to reveal wireless networking vulnerabilities by building a tool for compromising the overall security of a sensor network. The set of implemented attacks include some of the most severe existing routing attacks [5], like *Sinkhole*, *Replay attack*, *Selective Forwarding*, and HELLO *flood attack*, along with novel ones like *Malicious Code Injection*. Our contribution is twofold:

It is the first complete instance of such a tool that provides a number of attacks to be launched by an adversary, along with a framework for inspection and modification of data transmitted. Our goal is to describe the "best" ways to launch these attacks and demonstrate them in practise. We also present how memory related vulnerabilities can lead to injection and execution of arbitrarily long code in sensor devices following the Von Neumann architecture like Tmote Sky [15], Telos [16], EyesIFX [17]. This is achieved by exploiting buffer overflow leakages to smash the call stack and intrude a remote node over the radio channel. By sending a number of specially crafted packets, an adversary can inject a self-replicating worm that broadcasts itself and infects the network in a hop-by-hop manner.

Second, by publishing such an attack tool, we wish to shed light on revealing the weaknesses of the underlying protocols that are most widely used by sensor networks research community. We expect that our work will be particulary useful for demonstrating and educating users about the destructive impacts of various attacks and highlighting the need to come up with more efficient security protocols.

### 1.1 Paper Organization

The remainder of this paper is organized as follows. In Section 2, we list the ways that an adversary can compromise data confidentiality including carrier frequency, message size, message rate, and routing information along with a categorization of attacks that are supported by the tool. Section 3 is the heart of this work; it gives an overview of the tools' architecture along with a detailed presentation of all implemented system components. Description of all supported attacks is presented in Section 4. Finally, Section 5 concludes the paper.

## 2. NETWORK CONFIDENTIALITY THREATS AND WIRELESS ATTACKS

In wireless networking the overall security objectives remain the same as with wired networks: preserving confidentiality, ensuring integrity, and maintaining availability of information. Thus, identifying risks to sensor networks confidentiality posed by the availability of transactional data is extremely vital.

In an attempt to identify network confidentiality threats, we enhanced our attack tool with a network sniffer for overhearing network traffic (Section 3.1). In that way an adversary can process transmitted packets in order to extract vital information such as node IDs or traffic data. Our assertion is that traffic analysis can provide more information about a network's nodes and usage than simply decoding any data packet contents.

Pai *et al.* in [14] have outlined the ways that an adversary can compromise network confidentiality including *carrier frequency*, *message rate* and *size*, and *routing information.* The presented tool can use carrier frequency to launch a side-channel attack [18] in an attempt to identify the network's sensor hardware platform. An adversary could use either a spectrum analyzer or different sensor hardware in combination with our tool in order to detect the current communication frequency. Once the adversary discovers it, she can determine the hardware used and, thus, exploit all the protocol vulnerabilities arising from this specific platform.

This tool can also compromise a network's confidentiality by monitoring the rate and size of any transmitted/received messages. Specifically, the message rate can reveal information about the network application and the frequency of monitored events. This constitutes a severe threat since for some sensor applications, like health monitoring, it can lead to a violation of user's privacy. Furthermore, an adversary can examine the rate at which she overhears messages coming from a neighborhood and estimate the distance to the sensed event. Research has shown that the message reception rate increases when the distance to the event reporting node decreases.

Finally, overhearing routing information enables the network sniffing component to construct a directed graph of all neighboring nodes. Overheard packets flow along the edges of the graph revealing vital information about the underlying routing pattern. Observing this traffic pattern of a sensor network may deduce the location of the base station or other strategically located nodes. Furthermore, multihop communication routing protocols make it possible for an adversary to trace a stream of messages back to the information source.

Along with all the above threats, our tool can launch a number of attacks in an attempt to penetrate a sensor network's functionality. In order to achieve this, we have implemented a data stream framework for constructing and transmitting specially crafted packets. Most of the currently supported wireless attacks fall into one of the following categories:

- **Confidentiality Attacks**: These attacks attempt to

**Table 1: Supported Wireless Attacks**

| Type of Attack | Description |
|---|---|
| Eavesdropping | Capturing and decoding unprotected network traffic to obtain potentially sensitive information. |
| Data Replay | Capturing and/or *modifying* data frames for later replay. |
| Sinkhole | Lure as much network traffic as possible from a particular area. |
| Selective Forwarding | Intercept overheard packets and forward them *selectively* to the intended receiver. |
| Flooding | Sending forged HELLO (or *other* data) messages from random node IDs to cripple the network resources. |
| Program Image Dissemination | Send *new* program images to sensor nodes overwriting already existing ones. |
| Code Injection | Crafting and sending forged frames containing malicious code instructions. |

intercept private information sent over the wireless transmission medium.

- **Integrity Attacks**: These attacks send forged control, management or data frames to mislead the recipient or facilitate another type of attack.
- **Availability Attacks**: These attacks impede delivery of wireless messages to legitimate users by crippling the network resources.

Table 1 lists the specific attacks that can be launched by this attack tool (in its current version). A more detailed architectural description of the network attack tool components can be found in Section 3. In future work, we plan to enhance it by exploiting more network vulnerabilities and developing new kinds of attacks.

## 3. ATTACK TOOL ARCHITECTURE OVERVIEW

The attack tool is based on an intelligent component-based system. The hosted components are capable of monitoring any neighborhood traffic, decoding and logging overheard packets, constructing specially crafted messages and launching a number of attacks. Its core functionality is based on three main conceptual modules, as depicted in Figure 1:

- A **network sniffer** for passive monitoring and logging of radio packets. Any network traffic analysis or packet decoding can be done either in real time or offline through the implemented *packet description database*.
- A **network attack tool** that provides a number of actions for compromising a sensor network's security profile. It contains a *data stream framework* for constructing specially crafted packets that are transmitted by the *attack launcher* throughout the duration of an attack.
- A **network visualization** component that visualizes and displays the neighborhood topology, network traffic, node states and status of any performed attack.

The key design goal of this tool is its wide applicability; it should support passive inspection and compromise of a wide variety of sensor network protocols and applications.

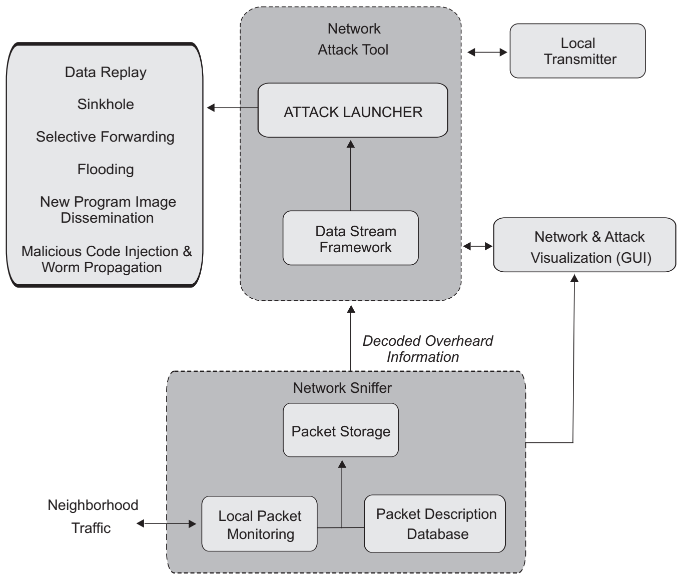
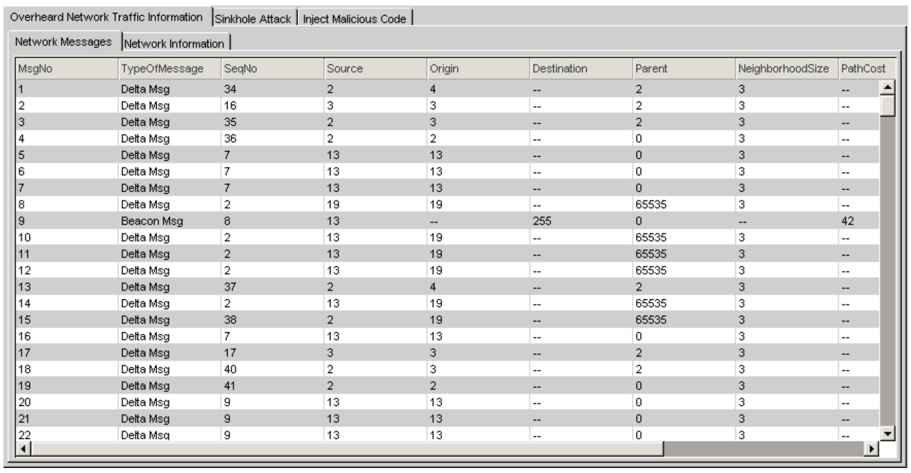


**Figure 1: Attack Tool Architecture Layout.**

By considering popular underlying protocols and message structures that are most widely used by the research community, we make the tool scalable and adaptive. When any raw network packets are available in the neighborhood, it uses them as the audit source in order to identify current used software versions and extract vital network information. While packet capture is performed in real time, traffic analysis can be done either online or offline. We believe that offline analysis provides a better way of monitoring and understanding a network's deployment. In what follows we give a more detailed description of the basic system components.

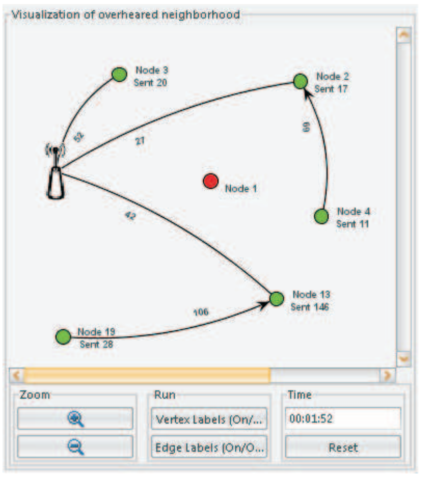
### 3.1 Network Sniffer Component

The network sniffer relies on packets that are overheard in a sensor's node neighborhood. It captures them and logs them for later analysis. Conceptually the sniffer consists of a *Local Packet Monitoring* module for gathering audit data to be forwarded, over its serial port, to the *Packet Storage* module for logging at the attached host. This allows offline analysis, through the *Packet Description Database*, in order to extract vital network information such as node IDs, traffic data or used protocol versions. Essentially, the sniffer enables the construction of a directed graph of all neighboring nodes. Overheard packets flow along the edges of the graph, as



| MsgNo | TypeOfMessage | SeqNo | Source | Origin | Destination | Parent | NeighborhoodSize | PathCost |
|---|---|---|---|---|---|---|---|---|
| 1 | Delta Msg | 34 | 2 | 4 | -- | 2 | 3 | -- |
| 2 | Delta Msg | 16 | 3 | 3 | -- | 2 | 3 | -- |
| 3 | Delta Msg | 35 | 2 | 3 | -- | 2 | 3 | -- |
| 4 | Delta Msg | 36 | 2 | 2 | -- | 0 | 3 | -- |
| 5 | Delta Msg | 7 | 13 | 13 | -- | 0 | 3 | -- |
| 6 | Delta Msg | 7 | 13 | 13 | -- | 0 | 3 | -- |
| 7 | Delta Msg | 7 | 13 | 13 | -- | 0 | 3 | -- |
| 8 | Delta Msg | 2 | 19 | 19 | -- | 65535 | 3 | -- |
| 9 | Beacon Msg | 8 | 13 | -- | 255 | 0 | -- | 42 |
| 10 | Delta Msg | 2 | 13 | 19 | -- | 65535 | 3 | -- |
| 11 | Delta Msg | 2 | 13 | 19 | -- | 65535 | 3 | -- |
| 12 | Delta Msg | 2 | 13 | 19 | -- | 65535 | 3 | -- |
| 13 | Delta Msg | 37 | 2 | 4 | -- | 2 | 3 | -- |
| 14 | Delta Msg | 2 | 13 | 19 | -- | 65535 | 3 | -- |
| 15 | Delta Msg | 38 | 2 | 19 | -- | 65535 | 3 | -- |
| 16 | Delta Msg | 7 | 13 | 13 | -- | 0 | 3 | -- |
| 17 | Delta Msg | 17 | 3 | 3 | -- | 2 | 3 | -- |
| 18 | Delta Msg | 40 | 2 | 3 | -- | 2 | 3 | -- |
| 19 | Delta Msg | 41 | 2 | 2 | -- | 0 | 3 | -- |
| 20 | Delta Msg | 9 | 13 | 13 | -- | 0 | 3 | -- |
| 21 | Delta Msg | 9 | 13 | 13 | -- | 0 | 3 | -- |
| 22 | Delta Msg | 9 | 13 | 13 | -- | 0 | 3 | -- |

(a)



(b)

**Figure 2: (a) Overheard packet content storage and visualization (b) Neighborhood topology shown by network sniffer.**

shown in Figure 2(b), and are provided with a number of operators for manipulating them.

Audit data consist of the communication activities within the sniffer's radio range. Such data can be collected by listening *promiscuously* to neighboring nodes' transmissions. By promiscuously we mean that when a node is within radio range, the local packet monitoring module can overhear communications originating from that node. Once captured by the radio, all packets are timestamped in order to facilitate subsequent time-based analysis. Timestamping is performed the moment the packet is received by the network sniffer.

Once the sniffer receives a packet, a flexible mechanism (due to lack of standardized protocols in sensor networks) is needed to decode overheard packets. That is why we have created the *Packet Description Database* which contains annotated message structures for the most widely used network protocols and applications (e.g., MintRoute [19] and MultihopLQI [20] routing protocols, Delta monitoring application, etc). This way, our packet decoder can use these loaded structures as a description of the overheard packet contents. The configuration of the packet description database is *extendable* and can be enhanced with new message structures. The user can specify message contents as C structs which will automatically be converted to message classes and be added to the underlying database. However, even in the case of an unrecognized overheard message, the sniffer still logs it and provides access and manipulating operators on the byte representation of its content. Thus, an adversary may alter it and resend it, leading again to other type of attacks like Replay, Selective Forwarding or even Denial of Service attacks.

All overheard packets are displayed by our network tool through the *Network Visualization* component, as illustrated in Figure 2(a). Message structure, packet contents and time of reception are provided to the user along with a number of operators for acting on them. These operators provide access, aggregation, alteration or re-transmission privileges for any of the stored messages. A more detailed description of the network visualization component can be found in Section 3.3.

## 3.2 Network Attack Tool Component

This component core functionality is to provide a number of actions for compromising the sensor network's security profile. After gathering audit data that are used by the network sniffer to extract vital information and identify the used sensor hardware platform and underlying protocols, a user can start launching a number of attacks (list of supported attacks can be found in Table 1).

The resulting network information stream from the packet decoder is fed to the *Data Stream Framework* of the attack tool component. This data stream processor uses the identified carrier frequency, message size and routing information as its configuration record. All these network characteristics are essential since they are used as the basis for any specially crafted message required by the *Attack Launcher*.

The attack launcher module is responsible for actually performing attacks like Data Replay, Sinkhole, Selective Forwarding, Flooding, Reprogramming and Code Injection. All these attacks either try to manipulate sensor data and functionality or affect the underlying routing topology. In any case, they allow an adversary to construct and transmit (periodically if necessary) messages with specially crafted content like fake sender ID, fake link quality or altered routing header. This is done by the data stream processor, which upon request from the attack launcher, constructs such a message and transmits it through the attached radio. A more detailed description of the implemented attacks, and the procedure that a user must follow in order to launch them, can be found in Section 4.

All the above described actions are handled by the user

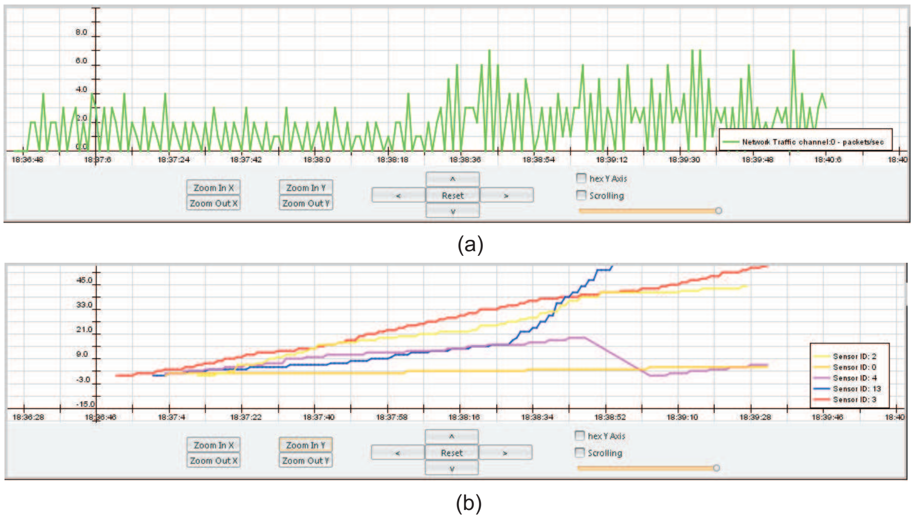


**Figure 3: Monitored traffic. (a) Overall neighborhood traffic. (b) Traffic overheard from each sensor node.**

through the graphical user interface provided by the network visualization component. Furthermore, attack status and additional information are displayed in a single log trace.

## 3.3 Network Visualization Component

The network visualization component shows, in real time, all the above information and the state of any performed attacks. The *neighborhood traffic*, *neighborhood topology* and *node states* are displayed in a friendly graphical user interface. For example, the overall neighborhood traffic (and traffic overheard by each sensor node) is monitored by continuous graphs, as shown in Figures 3(a) and (b) respectively.

The core abstraction implemented by this user interface is a neighborhood graph, where nodes and links can be annotated with supportive information like node IDs, link quality, routing parent, etc. A snapshot of such a topology is shown in Figure 2(b). Here, node color indicates functionality type (green: network node, red: sensor hardware attached to the attack tool). The antenna represents the central base station of the network. Edges between the nodes indicate network links, while the numbers above the edges indicate the quality of the link (LQI), as this is produced by the underlying routing protocol. Nodes can be selected to display a textual summary of the information gathered about them.

One of the most important pieces of information displayed by the user interface involves the routing path taken by a packet travelling from a source node $s$ to a destination $d$. This routing state is inferred by observing packet transmissions produced by the routing protocol. Arcs indicate the paths that multi-hop data messages follow.

## 4. IMPLEMENTED ATTACKS & ACTIONS

Many sensor network deployments are quite simple, and for this reason they can be even more susceptible to attacks. What makes it particularly easy for attackers is the fact that most protocols are not designed having security threats in mind. As a consequence, they rarely include security protection and little or no effort is usually required from the side of an adversary to perform an attack. So, it is very important to study realistic attacker models and evaluate their practicality and effectiveness through a tool as the one presented in this work.

The nature of wireless network communications opens the way to four basic attacks: Interception, Alteration, Disruption and Code or Packet Injection [5]. Most network layer attacks against such networks fall into one of these categories. Our attack tool (in its current version) gives the user the opportunity to perform, in addition to eavesdropping and sniffing, the following actions:

**Data Replay.** A replay attack is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated. As the adversary is capable of listening to any message transmitted over the network medium, she may insert "new" messages or manipulate any message sent by a legitimate participant of the network. In the presented tool, all overheard messages are stored into the *Packet Description Database*, so the user is able to change them and re-transmit them at a later time.

**Sinkhole Attack.** The sinkhole attack [5, 10] is a particularly severe attack that prevents the base station from obtaining complete and correct sensing data, thus forming a serious threat to higher-layer applications. Using this attack, an adversary can attract nearly all the traffic from a particular area. Typically, sinkhole attacks work by making a malicious node look especially attractive to surrounding nodes with respect to the underling routing algorithm. Our motivation for mounting sinkhole attacks is that it makes other kind of attacks, like Selective Forwarding, trivial.

**Selective Forwarding.** In a selective forwarding attack, an adversary may refuse to forward certain messages and simply drop them, ensuring that they will not be propagated any further. This attack is especially effective if combined with an attack that gathers network traffic and can be used as an attack vector to mount denial of service attacks.

**Flooding.** In a HELLO flood attack, an attacker can send or replay HELLO messages with high transmission power. In this way, she creates an illusion of being a neighbor to many nodes and can disrupt the construction of the underlying routing tree, facilitating further types of attacks.

**Malicious Code Injection.** By taking advantage of memory related vulnerabilities in sensor nodes, like buffer overflows, an adversary may send crafted packets to trigger a stack overflow and execute arbitrary code on the target system [13]. She may also create and send a self-replicating worm that broadcasts itself and infects the network in a hop-by-hop manner in order to completely take control of it, shut the network down or change its functionality.

**Node Ping Operator & Program Image Dissemination.** These operations are provided by the Deluge over-the-air programming protocol [21]. The *ping action* sends a message to a specific sensor node to request about its state, its currently executing program image and what other images are stored in that node. *Program Image dissemination* is a fundamental service in sensor networks that relies upon reliable broadcast of image updates. However, it faces threats since an adversary may easily subvert it by modifying or replacing the real code image being propagated to sensor nodes.

In what follows, we give an overview of the procedures that should be followed by a user in order to perform the above described attacks. Since it is not in the scope of this work to present in detail the attack techniques and clarify their effectiveness (the reader is referred to the original papers), we confine ourselves to highlight the steps and actions performed by the attack tool throughout the duration of the attack.

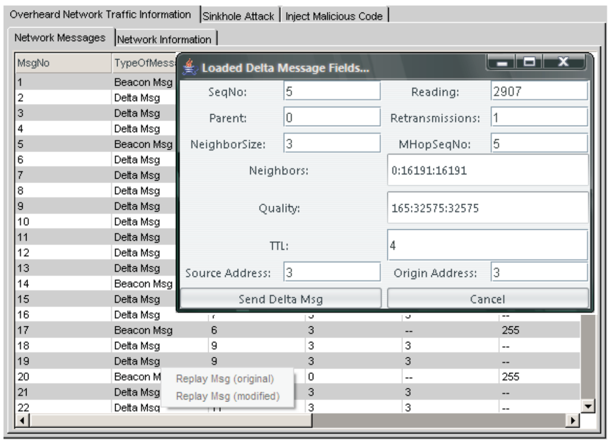
## 4.1 Attacks Walkthrough

### *4.1.1 Data Replay, Selective Forwarding and HELLO Flooding Attack*

As we described in Section 3.1, all overheard messages are decoded, stored and displayed by our network tool sniffer, as illustrated in Figure 2(a). Message structure and packet contents are provided to the user along with a number of operators for manipulating them.

In the case of data replay, all captured packets may be retransmitted by the user at a later time. The attack tool enables transparent data access and alteration upon selection of a logged displayed message. As depicted in Figure 4, once a message is selected, the user is presented with two options: replay the *original* message or replay an *altered* version of it. If the first case, a copy of the selected message is fed to the *Attack Launcher* by the *Packet Storage* module. Then, it is transmitted using the attached radio. In the second case, a user interface is provided that gives the user the ability to alter the message contents before transmission (Figure 4). If the selected message is of an unrecognized structure (could not be found in the *Packet Description Database*), the user interface provides the byte representation of its contents. Thus, the user can still change it and re-transmit it.

As far as Selective Forwarding is concerned, our tool can



| MsgNo | TypeOfMess | | | | |
|---|---|---|---|---|---|
| 1 | Beacon Msg | | | | |
| 2 | Delta Msg | | | | |
| 3 | Delta Msg | | | | |
| 4 | Delta Msg | | | | |
| 5 | Beacon Msg | | | | |
| 6 | Delta Msg | | | | |
| 7 | Delta Msg | | | | |
| 8 | Delta Msg | | | | |
| 9 | Delta Msg | | | | |
| 10 | Delta Msg | | | | |
| 11 | Delta Msg | | | | |
| 12 | Delta Msg | | | | |
| 13 | Delta Msg | | | | |
| 14 | Beacon Msg | | | | |
| 15 | Delta Msg | | | | |
| 16 | Delta Msg | 7 | 3 | 3 | -- |
| 17 | Beacon Msg | 6 | 3 | -- | 255 |
| 18 | Delta Msg | 9 | 3 | 3 | -- |
| 19 | Delta Msg | 9 | 3 | 3 | -- |
| 20 | Beacon M | | 0 | -- | 255 |
| 21 | Delta Msg | | 3 | 3 | -- |
| 22 | Delta Msg | 11 | 3 | 3 | -- |

Loaded Delta Message Fields...

| | | | |
|---|---|---|---|
| SeqNo: | 5 | Reading: | 2907 |
| Parent: | 0 | Retransmissions: | 1 |
| NeighborSize: | 3 | MHopSeqNo: | 5 |
| Neighbors: | 0:16191:16191 | | |
| Quality: | 165:32575:32575 | | |
| TTL: | 4 | | |
| Source Address: | 3 | Origin Address: | 3 |

Send Delta Msg — Cancel

**Figure 4: Replay original or altered overheard logged messages.**

be thought as part of the existing sensor network since it is connected to a sensor platform like the one used in the deployed network. Thus, the user can select which of the received displayed messages will actually be forwarded by the tool. This will ensure that they will not be propagated any further, possibly leading to other types of attacks like Denial of Service of *Black Hole* [5] attacks.

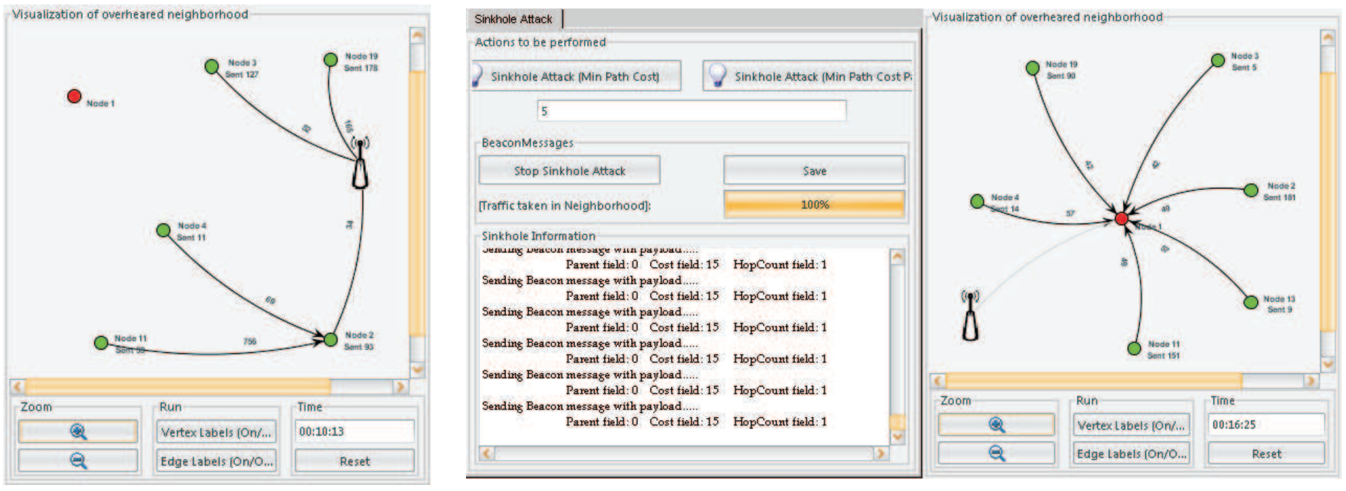
Finally, HELLO flooding attempts to attack the underlying routing tree. It requires the attacker to broadcast specially crafted HELLO packets for advertising possible fake IDs to other network nodes. This is achieved by a single hop transmission using the attached radio with enough power to reach every sensor node. Eventually, this "flooding" can lead to other type of attacks like one-way *Wormholes* [5].

### *4.1.2 Sinkhole Attack*

In a sinkhole attack, a malicious node tries to draw all or as much traffic as possible from a particular area by making itself look attractive to the surrounding nodes with respect to the underlying routing metric. There appears to be a great diversity in deployed routing protocols for sensor networks. However, most of them, use link quality calculations as the routing cost metric to build the routing tree towards the base station. Such routing protocols, like MintRoute [19] and MultihopLQI [20], are supported by the presented tool.

In such protocols, each node broadcasts a *beacon* message and the receivers extract the link quality (LQ) based either on their radio chip or on the *packet loss* of the packets received from this neighbor. The most attractive link is selected for transmission and is the one with the best link quality. According to this algorithm, the goal of sinkhole attack is to advertise a very good LQ in order for all neighboring nodes to choose the tools' attached node as their parent. More details about the strategies that the tool follows to successfully launch such an attack along with a performance evaluation can be found in [10].

Eventually this is achieved using a periodic transmission of specially crafted "routing packets" (beacons). The time interval between the transmissions is configurable and given



```
Sending Beacon message with payload.....
                Parent field: 0   Cost field: 15   HopCount field: 1
Sending Beacon message with payload.....
                Parent field: 0   Cost field: 15   HopCount field: 1
Sending Beacon message with payload.....
                Parent field: 0   Cost field: 15   HopCount field: 1
Sending Beacon message with payload.....
                Parent field: 0   Cost field: 15   HopCount field: 1
Sending Beacon message with payload.....
                Parent field: 0   Cost field: 15   HopCount field: 1
Sending Beacon message with payload.....
                Parent field: 0   Cost field: 15   HopCount field: 1
```

(a) (b)

**Figure 5: Launching the Sinkhole Attack on a real deployed network (a) The neighborhood topology before the attack (b) The neighborhood topology after the attack.**

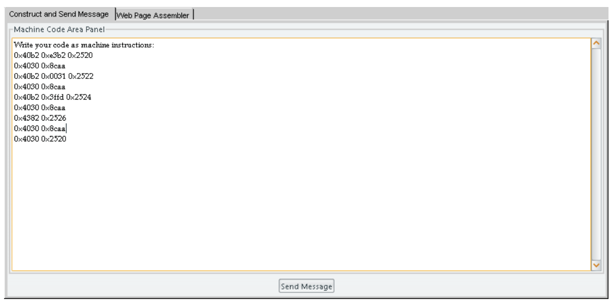
by the user upon initiation of the attack. Then, the *Data Stream Framework* module takes over to construct and transmit these packets. For an illustration, following Figure 5(a), the attached node (ID=1) tries to convince its neighboring nodes to choose it as their "parent". The result is shown in Figure 5(b). Let us note here that by trying different network topologies, we rarely missed attracting 100% of the nodes.

### 4.1.3 *Malicious Code Injection*

Malicious code (or malware) is defined as a software designed to execute attacks on software systems and fulfill the harmful intents of an attacker. The presented tool is capable of injecting such malware on wireless sensor nodes that are based on the Von Neumann architecture. This is achieved by exploiting a buffer overflow vulnerability to smash the call stack and intrude a remote node over the radio channel. A more detailed demonstration of how to execute malware on sensor embedded devices and how it can be crafted to become a self-replicating worm, that broadcasts itself, can be found in [13].

In our case, blocks of the attack code are sent as data payload of a specially crafted message. The goal of each injected packet is to copy data (malicious instructions) into the sensors' memory space. These instructions are represented by unique 2-byte op-codes in order to be decoded by the CPU. As illustrated in Figure 6, a user can write the byte representation of his/her code instructions in a special user interface. In this example, the (malicious!) functionality of the code is to toggle the LEDs of a sensor node.

Upon start up, the *Data Stream Framework* module "reads" the given instructions and computes the number of needed messages to carry the attack code. Then, it constructs these specially crafted packets by adding code blocks as their content and transmitting them in a sequential order. Upon completion, it informs the user by printing an appropriate message. Let us note here that the attack tool provides ac-



```
Write your code as machine instructions:
0x40b2 0xe3b2 0x2520
0x4030 0x8caa
0x40b2 0x0031 0x2522
0x4030 0x8caa
0x40b2 0x3ffd 0x2524
0x4030 0x8caa
0x4382 0x2526
0x4030 0x8caa
0x4030 0x2520
```

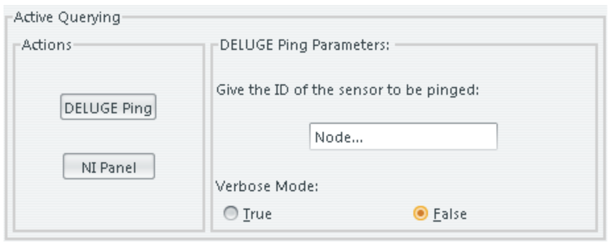
**Figure 6: Malicious code to be injected in the network.**

cess to an online *assembler* and *disassembler* in order for the user to be able to construct the byte representation of his/her attack code in a friendly manner.

### 4.1.4 *Program Image Dissemination & Ping Operation*

Deluge is a popular data dissemination protocol for program images over a multihop sensor network. In the presented tool, we have incorporated the two basic functionalities provided by this over-the-air-programming mechanism which are *program image dissemination*, through epidemic propagation, and *ping operation*.

Conceptually, the tool is fed with a compiled program image which is then divided into *pages*, each consisting of $N$ packets. Then, the *Data Stream Framework* starts "advertising" this new image and upon request it transmits, in a sequential order, all page packets using the attached radio. More information about how Deluge works can be found in [21]. In this way, the user can change the network's functionality by re-programming all nodes with his/her program code. This may lead to other types of attacks like Denial of Service.


Active Querying

Actions

DELUGE Ping

NI Panel

DELUGE Ping Parameters:

Give the ID of the sensor to be pinged:

Node...

Verbose Mode:

True [✓ False]

**Figure 7: Pinging a node for software information.**

One other feature of our attack tool that can be used in compromising a network's confidentiality is the *ping* operation. By pinging a node, you actually request information regarding its currently installed software version. The ping response will display information about the executing program image and other images that are stored on this node. As illustrated in Figure 7, a user can ping each one of the overheard neighboring nodes by adding the appropriate ID. In this way, he/she can can extract vital information about the network application [22].

## 5. CONCLUSIONS

In this paper, we have identified some of the sensor networks vulnerabilities that can be exploited by an attacker for launching various kinds of attacks. We have demonstrated the practicality of these attacks by building an *attack tool* for compromising the network's confidentiality and functionality. The results of this work serve a three-fold purpose: to reveal the vulnerabilities of such networks, to study the effects of severe attacks on the network itself and to motivate a better design of security protocols that can make them more resilient to adversaries. Wireless sensor network security is an important research direction and tools like the current one may be used in coming up with even more attractive solutions for defending these types of networks.

## 6. REFERENCES

[1] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97, New York, NY, USA, 2002. ACM.

[2] J. Tateson, C. Roadknight, A. Gonzalez, T. Khan, S. Fitz, I. Henning, N. Boyd, C. Vincent, and I. Marshall. Real World Issues in Deploying a Wireless Sensor Network for oceanography. In *Workshop on Real-World Wireless Sensor Networks REALWSN'05*, Stockholm, Sweden, June 2005.

[3] D. Trossen, D. Pavel, G. Platt, J. Wall, P. Valencia, C. A. Graves, M. S. Zamarripa, V. M. Gonzalez, J. Favela, E. Livquist, and Z. Kulcs Sensor networks, wearable computing, and healthcare applications. *IEEE Pervasive Computing*, 6:58–61, 2007.

[4] A. Becher, Z. Benenson, and M. Dornseif. Tampering with motes: Real-world physical attacks on wireless sensor networks. In J. A. Clark, R. F. Paige, F. Polack, and P. J. Brooke, editors, *SPC*, volume 3934 of *Lecture Notes in Computer Science*, pages 104–118. Springer, 2006.

[5] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *AdHoc Networks Journal*, 1(2–3):293–315, September 2003.

[6] I. Krontiris, Z. Benenson, T. Giannetsos, F. C. Freiling, and T. Dimitriou. Cooperative intrusion detection in wireless sensor networks. In *EWSN '09: Proceedings of the 6th European Conference on Wireless Sensor Networks*, pages 263–278, Berlin, Heidelberg, 2009. Springer-Verlag.

[7] A. Perrig, J. Stankovic, and D. Wagner. Security in wireless sensor networks. *Communications of the ACM*, 47(6):53–57, 2004.

[8] V. Bhuse and A. Gupta. Anomaly intrusion detection in wireless sensor networks. *J. High Speed Netw.*, 15(1):33–51, 2006.

[9] G. Li, J. He, and Y. Fu. Group-based intrusion detection system in wireless sensor networks. *Comput. Commun.*, 31(18):4324–4332, 2008.

[10] I. Krontiris, T. Giannetsos, and T. Dimitriou. Launching a sinkhole attack in wireless sensor networks; the intruder side. *Wireless and Mobile Computing, Networking and Communication, IEEE International Conference on*, 0:526–531, 2008.

[11] D. R. Raymond and S. F. Midkiff. Denial-of-service in wireless sensor networks: Attacks and defenses. *IEEE Pervasive Computing*, 7:74–81, 2008.

[12] A. Francillon and C. Castelluccia. Code injection attacks on harvard-architecture devices. In *15th ACM Conference on Computer and Communications Security (CCS)*, Alexandria, VA, USA, 2008.

[13] T. Giannetsos, T. Dimitriou, I. Krontiris, and N. R. Prasad. Arbitrary code injection in sensor networks through self-propagating worms in von neumann architecture devices. *Accepted to be published in the Computer Oxford Journal for the Algorithms, Protocols, and Future Applications of Wireless Sensor Networks special issue*, 2010.

[14] S. Pai, M. Meingast, T. Roosta, S. Bermudez, S. B. Wicker, D. K. Mulligan, and S. Sastry. Transactional confidentiality in sensor networks. *IEEE Security and Privacy*, 6(4):28–35, 2008.

[15] Tmote Sky Quick Start Guide. Technical report.

[16] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 48, Los Angeles, California, 2005.

[17] V. Handziski, J. Polastre, J.-H. Hauer, and C. Sharp. Flexible hardware abstraction of the TI MSP430 microcontroller in TinyOS. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 277–278, Baltimore, MD, USA, 2004.

[18] C. C. Tiu and C. C. Tiu. A new frequency-based side channel attack for embedded systems. master degree thesis, deparment of electrical and computer engineering,university of waterloo,waterloo. Technical report, 2005.

[19] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Wireless sensor networks for structural health monitoring. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 427–428, 2006.

[20] TinyOS. `http://www.tinyos.net/tinyos-1.x/tos/lib/MultiHopLQI`, 2004.

[21] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 81–94, New York, NY, USA, 2004. ACM.

[22] I. Krontiris and T. Dimitriou. Authenticated in-network programming for wireless sensor networks. In *Proceedings of the 5th International Conference on AD-HOC Networks & Wireless (ADHOC-NOW '06)*, pages 390–403, 2006.