MULTI-VECTOR PORTABLE INTRUSION DETECTION SYSTEM

Benjamin Robert Moyers

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
In
Computer Engineering

Dr. Joseph G. Tront, Committee Chair
Mr. Randolph C. Marchany
Dr. Patrick R. Schaumont

July 30, 2009

Blacksburg, VA

Keywords: Intrusion Detection, Bluetooth Security,
Wireless Security, Mobile Device Security.

MULTI-VECTOR PORTABLE INTRUSION DETECTION SYSTEM

Benjamin Robert Moyers

ABSTRACT

This research describes an intrusion detection system designed to fulfill the need for increased mobile device security. The Battery-Sensing Intrusion Protection System (B-SIPS) [1] initially took a non-conventional approach to intrusion detection by recognizing attacks based on anomalous Instantaneous Current (IC) drainage. An extension of B-SIPS, the Multi-Vector Portable Intrusion Detection System (MVP-IDS) validates the idea of recognizing attacks based on anomalous IC drain by correlating the detected anomalies with wireless attack traffic from both the Wi-Fi and Bluetooth mediums. To effectively monitor the Wi-Fi and Bluetooth mediums for malicious packet streams, the Snort-Based Wi-Fi and Bluetooth Attack Detection and Signature System (BADSS) modules were introduced.

MVP-IDS illustrates that IC anomalies, representing attacks, can be correlated with wireless attack traffic through a collaborative and multi-module approach. Furthermore, MVP-IDS not only correlates wireless attacks, but mitigates them and defends its clients using an administrative response mechanism.

This research also provides insight into the ramifications of battery exhaustion Denial of Service (DoS) attacks on battery-powered mobile devices. Several IEEE 802.11 Wi-Fi, IEEE 802.15.1 Bluetooth, and blended attacks are studied to understand their effects on device battery lifetimes. In the worst case, DoS attacks against mobile devices were found to accelerate battery depletion as much as 18.5%. However, if the MVP-IDS version of the B-SIPS client was allowed to run in the background during a BlueSYN flood attack, it could mitigate the attack and preserve as much as 16% of a mobile device's battery lifetime as compared with an unprotected device.

ACKNOWLEDGEMENTS

First, I would like to thank Mr. Randy Marchany for his guidance, mentorship, and patience throughout the duration of graduate studies with both myself and all of the research setbacks I encountered. He was always there to suggest new ideas and was a continual source of encouragement.

I also wish to extend my sincerest appreciation to Mr. Wayne Donald and Mr. Randy Marchany for allowing me to use the Virginia Tech IT Security Laboratory. If not for this nurturing research environment, this research effort would not have been possible. In addition, I would like to thank the SANS Institute for the funding provided to purchase needed equipment used in the lab.

I would also like to thank my advisor, Dr. Joseph Tront, for his support and guidance in shaping my research studies, as well as Dr. Patrick Shaumont for serving on my committee.

I also wish to extend my utmost gratitude to Col. Timothy Buennemeyer for not only taking me under his wing as a research colleague and friend, but for also allowing me to use his dissertation work as a starting point for my research. Furthermore, I would like to thank John Paul Dunning for his countless time and energy spent in helping me on this project and also in turning feasible ideas into working systems. I would also like to thank Mr. Brad Tilley for taking the time to edit and review this entire document. It was always helpful to have another set of eyes to spot minor mistakes and to always give useful feedback.

Many thanks also needs to be given to the my fellow researchers in the IT Security Lab; Beau Frazier and Lee Clagett. They were always a positive source of creative ideas and helped in many ways to contribute to this research effort.

Last, but certainly not least, I would like to thank my family and friends for always being a constant source of motivation and support for me to finally finish my degree. Without them, this research endeavor and degree would most certainly be unobtainable. Thank you all!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

Personal Digital Assistants (PDAs) and smart phones, also known as Portable Information Devices (PIDs), are less computationally powerful than desktop and laptop Personal Computers (PCs), but possess many of the same features and allow for much of the same functionality. Two defining features included in PIDs are the IEEE 802.11 (Wi-Fi) and IEEE 802.15.1 (Bluetooth) capabilities. Though they are similar, many basic security measures common in PCs are not present in PIDs, primarily because of limited power and computation cycle resources. This research shows that the addition of an Intrusion Detection System (IDS) on PIDs can greatly enhance their security.

This research addresses mobile device security and extends the original Battery-Sensing Intrusion Protection System (B-SIPS) [1] design by introducing the Multi-Vector Portable - Intrusion Detection System (MVP-IDS). MVP-IDS validates reported anomalous battery depletion from B-SIPS clients with real-time Wi-Fi and Bluetooth traffic using attack signature detection modules. To correlate instantaneous current (IC) anomalies with Wi-Fi and Bluetooth attack traffic, MVP-IDS integrates B-SIPS anomaly detection with the signature-based matching systems of Snort [2] and a newly developed research system, Bluetooth Attack Detection and Signature System (BADSS).

Section 1.1 provides motivation to protect PIDs in the medical, business, military, and government sectors. Section 1.2 follows, giving a brief overview of the B-SIPS methodology and conceptual goals. Section 1.3 presents a brief overview of the MVP-IDS methodology. Section 1.4 lays out the improvements and extensions to the B-SIPS that MVP-IDS achieves. Section 1.5 summarizes the thesis organization.

1.1 Motivation

While not all of the devices used in this motivation are PIDs, the purpose is to show that protecting mobile wireless communication devices should be an essential design constraint. In the medical industry, new devices are allowing patients enhanced health care through the utilization of implantable devices. These devices can monitor the physiological conditions within the body and relay information to remote monitoring locations for data analysis and device recalibration[3]. Some of these implantable devices include cardiac pacemakers, defibrillators, drug delivery systems, hearing-aids, epileptic brain monitors, and neurostimulators [3] [4]. Another application of wireless communication in the medical field is the use of Bluetooth-enabled computer chips embedded into prosthetic limbs to control joint movement [5]. While the development of medical devices to increase patient health is a great application of technology, the security of the technology must also be of paramount concern. Computer scientists have shown that eavesdropping on signals and extracting information

from these devices is possible. Even worse, they have shown that they are able to gain wireless access to a heart defibrillator/pacemaker combo and reprogram the device to send fatal jolts of electricity to the patient [6] [7].

Small and large businesses alike are using PIDs to increase productivity, allowing employees access to company information or clients even when they aren't in the office. Many businesses are turning to PIDs running BlackBerry business/enterprise solutions, OSX, Symbian OS, and Windows Mobile to satisfy their mobile needs. PCs in the office usually follow a strict protocol regarding security; most don't even allow users administrative privileges to their own PCs for security reasons. However, PIDs in these environments have an increased risk of exploitation due to their lack of security, but most businesses don't protect them as if they are an increased security risk. The National Vulnerability Database has 14 vulnerabilities for BlackBerry smart phones alone [8]. PIDs, like PCs, can be infected with spyware or viruses, text messages and email can be intercepted, and microphones can be remotely turned on to record and/or monitor conversations [8]. An attack on a business PID could leak confidential company information, client information, and even possibly cripple device communication if a Denial of Service (DoS) attack is used.

The military is increasingly relying on mobile devices to more efficiently move troops and keep soldiers communicating with other soldiers, tanks, aircraft, and ships in battle. U.S. Navy personnel are using PIDs as electronic clipboards to take inventory, as well as the army using them for bar code scanning during logistics and supply chain management [9]. The U.S Army also has a Land Warrior program that has infantrymen using PIDs to complete missions and distribute combat information. However, if these wireless links transmitting classified information between soldiers become compromised, many lives could be in danger. Also, devices used in military applications with global position system (GPS) capabilities are susceptible to tracking and could inform enemy intelligence of troop and/or equipment movement. While the concept of keeping soldiers connected to base stations during combat through the use of PIDs is an extraordinary idea, securing them must also be a top priority.

Government officials have a high need to conduct business on the move, but doing so under intense security scrutiny is difficult. While presidents before him were not intent on using smart phones in their daily lives, President Barak Obama feels as if using a BlackBerry is an essential part of his everyday routine [8]. However, with the many vulnerabilities and exploits against BlackBerry's, Greg Shipley, chief technology officer at Neohapsis, a government risk and compliance consultancy, considers a government official using a BlackBerry "a risky proposition". Because of vulnerabilities in commercial off-the-shelf PIDs, the U.S National Security Agency (NSA) contracted two companies to develop smart phones for use by government employees requiring "Type I" security [8] [10]. Currently, the NSA has certified two PIDs as being acceptable for Top Secret use, the Sectera Edge from General Dynamics and the Guardian from L-3 Communications. Funded by the Defense Department, both of these devices were created through the Secure Mobile Environment Portable Electronic Device Project with the goal of allowing government officials mobile convenience while maintaining secure communications. Attacks against

these devices could not only compromise classified information, but could also place national security and other government officials in danger as well.

1.2 B-SIPS Overview

This thesis extends and validates the previous B-SIPS research endeavor [1] which attempted to bring intrusion and attack detection to PIDs. Conventional IDS software, such as Snort and Norton Internet Security, would consume valuable battery and processing resources that PIDs simply do not have to spare. B-SIPS presented a viable, net-centric solution for securing mobile devices in malicious Wi-Fi and Bluetooth environments while also preserving battery life. The original B-SIPS design consisted of two core components, the B-SIPS client and the Correlation Intrusion Detection Engine (CIDE) Server. The B-SIPS client is a host-based application that uses a Dynamic Threshold Calculation (DTC) algorithm to continually monitor device operating characteristics for anomalous IC changes [1]. Malicious wireless communications, via Wi-Fi or Bluetooth, produce anomalous IC expenditures, therefore allowing the client to detect attacks. Once a possible attack is detected, the client sends battery information to the CIDE server for further analysis and data mining. The CIDE server provides many forensic analysis tools for security administrators (SAs) to visualize information and determine if the anomalous IC drain was in fact an attack, or a false alarm caused by a normal process using an increased anomalous amount of IC. With this being a simplified discussion of the B-SIPS system, an elaboration of B-SIPS details and inner workings can be found in Chapter 2.

1.3 MVP-IDS Design and Methodology

The main objective of this research is to hinder outside sources from negatively influencing the usability and lifetime, per battery charge, of PIDs. Since PIDs are dependent on mobile battery sources with limited lifetimes, attacks focused on draining battery life can, in effect, produce a DoS on these devices [1, 11, 12].

The methodology behind MVP-IDS is simple: recognize a significant change in IC on a PID and correlate the change with malicious Wi-Fi or Bluetooth traffic. To accomplish this, MVP-IDS is divided into four distinct modules: *B-SIPS client* for IC anomaly triggering, *Snort-Based Wi-Fi module* for Wi-Fi attack detection, *BADSS module* for Bluetooth attack detection, and *CIDE Server [1]* for attack correlation and response, as shown in Figure 1.

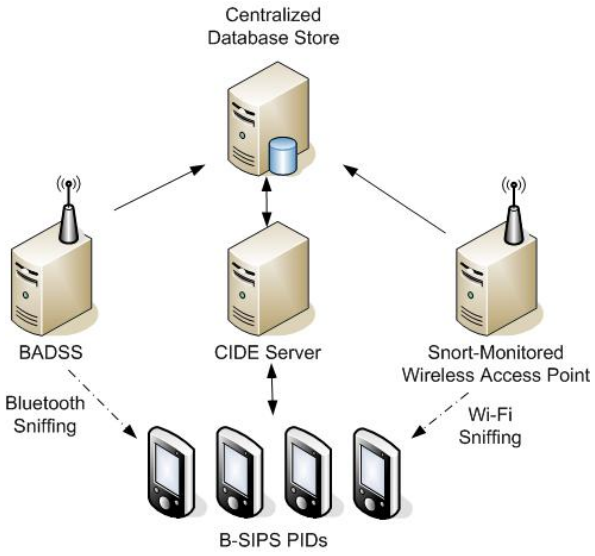


Figure 1. MVP-IDS system overview.

1.3.1 B-SIPS Client

B-SIPS client attack detection is based on irregularities in device IC changes. B-SIPS clients poll the smart battery for voltage, current, temperature, percent battery life, battery flag, and AC line status to determine battery consumption status. The DTC algorithm was developed to deter the system from falsely identifying IC changes as actual attacks by considering known reasons for battery drain, including the starting of known processes and changing in device back lighting. Based on this information, alert thresholds are adjusted to reduce false positives [1].

1.3.2 A Theoretical Approach to Packet Capturing on PIDs

A theoretical approach to detecting Wi-Fi and Bluetooth attacks is to monitor all traffic sent and received from a PID through those mediums and trigger alarms based on known malicious packets. This task could be accomplished by the use of raw sockets in an operating system (OS), allowing access to the header and payload of all transmitted packets [13]. Our research first proposed to monitor all wireless traffic for PIDs in this fashion and send the captured traffic back to the CIDE server to be analyzed by Snort and BADSS. However, since sending every packet in its entirety back to the CIDE server would double the amount of typical network traffic and was not completely necessary for attack detection, it was decided to send only portions of each Wi-Fi packet, along with high-level signaling packets for Bluetooth.

1.3.3 Snort-based Wi-Fi Module

Unfortunately, the ready availability of raw sockets for Wi-Fi was removed from the Windows Mobile environment by Microsoft for security reasons, leaving us to pursue hypothetical results by using a simulation to validate packet capturing for attack determination. In our simulation, packets are captured as they pass through the Snort monitored Wireless Access Point (WAP) with which PIDs are associated. While not the ideal mobile solution for capturing traffic, this approach provides more accurate results and will not be limited by the amount of captured data that can be transferred without consuming vast amounts of PID resources.

1.3.4 BADSS Module

BADSS was originally designed to detect Bluetooth attacks in the same fashion as originally proposed in the theoretical approach to packet capturing. Capturing of all Bluetooth traffic for the PID allows communication monitoring in real-time, with correlation done by the CIDE server. Due to the lack of Bluetooth raw sockets, the Merlin II Bluetooth protocol analyzer [14] was used to passively capture traffic between communicating Bluetooth devices. BADSS parses textual capture files from the analyzer and attempts to correlate the recorded traffic with known attack signatures in the BADSS signature database. Once detected, attacks are logged in the BADSS attack database for analysis and viewing by the CIDE server.

1.3.5 CIDE Server

The CIDE server functions as the supervisor for the system, performing attack correlation and developing grounds for administrative action. The correlation and administrative analysis is done external to the PID by design due to limited memory, battery power, and processing constraints of PIDs. The CIDE server communicates with the Snort and BADSS modules to correlate which attack vector(s) triggered a DTC breach by a B-SIPS client.

Once the CIDE server has information regarding the correlation of an IC anomaly with an associated attack signature, it then sends administrative responses back to the attacked PID. Administrative responses not only tell the PID user that they are being attacked, but also supply details about the attack and administrative actions being taken.

1.4 Significant MVP-IDS Improvements

The following eight significant improvements have been added to the B-SIPS design to further extend the project and create MVP-IDS. Through the utilization of these improvements, MVP-IDS is more robust and better equipped to secure PIDs in malicious environments.

1. A pattern-matching packet-based Bluetooth IDS has been added for Bluetooth attack recognition.
2. A Snort-monitored WAP scans Wi-Fi traffic and logs attacks to a CIDE-viewable database.
3. With the use of Snort for Wi-Fi traffic and BADSS for Bluetooth traffic, differentiating attacks and attack mediums is now possible.
4. Recognition of blended, or multi-vector attacks, which incorporate attacks from both Wi-Fi and Bluetooth, is now possible with MVP-IDS.
5. With Snort and BADSS logging attacks to a centralized database, the CIDE server is now able to conduct real-time correlation from B-SIPS clients as soon as they report an attack.
6. By design, the B-SIPS project only allowed for client to server communication. MVP-IDS now has the ability to actively respond to attacks using bi-directional communication.
7. When an attack was detected by the original B-SIPS implementation, both the Wi-Fi and Bluetooth interfaces were disabled to mitigate attacks. With the MVP-IDS attack medium differentiation system, only wireless interfaces deemed under attack are disabled.
8. To mitigate tampering of data transactions between B-SIPS clients and the CIDE server, confidentiality, authentication, message integrity, and non-repudiation have been integrated into all message passing.

1.5 Thesis Organization

The remainder of this document is organized as follows. Chapter 2 provides a brief overview of background material and ideas used in this work. Chapter 3 presents related work from past research publications. Chapter 4 details the MVP-IDS design and methodology, showing how each component of the system interacts to provide a secure environment for PIDs to function. Chapter 5 explores the MVP-IDS test bed, testing procedures, and gives an in-depth analysis of the results. Chapter 6 provides a summary of the research and suggests ideas for future work.

2. Background

One must fully grasp the underlying protocols and ideas used in MVP-IDS' design to allow for a more complete understanding of its details and intricacies. This chapter intends to introduce background material used in this research attempt for improved PID security. Section 2.1 presents defining characteristics of PDAs and smart phones. Section 2.2 introduces the ideas of confidentiality, authentication, message integrity, and non-repudiation, as well as describes the need for all in the attempt to secure data communications. Section 2.3 gives a brief introduction to the TCP/IP suite and the use of it in Wi-Fi environment. Section 2.4 delves into the Bluetooth protocol and its features. Section 2.5 explores the idea of intrusion detection and variety of ways it can be implemented.

2.1 PDAs and Smart Phones

2.1.1 PDAs

PDAs were first introduced in the mid 1990's, with the first popular models being the Apple Newton and the Palm Pilot. The simplified definition of a PDA is: "a computer that fits in the palm of your hand [15] [16]." Many features that come standard on most PDAs include:

- **LCD Displays:** The majority of the surface area of the device is used as a graphical interface by which users interact via a stylus, or finger taps. This allows users to easily and effortlessly navigate between menus to perform desired tasks.

- **Operating systems:** Usually more basic and downsized versions than installed on PCs, these OS versions function in much the same way that full featured OSs do. A few of the popular OSs included Symbian, Palm OS, and Windows Mobile.

- **Synchronization:** Most users tend to need PDAs for mobile management of appointments, contact information, email, and daily planning. With this data usually needed on both a PDA and PC, synchronization between the two is essential. PDAs can be connected, or synced, with PCs, allowing the user to keep data, e-mail, etc., current on both devices. This is done so that each device reflects accurate and up-to-date information.

- **Software Applications:** PDAs allow for data processing and on-the-go productivity enhancement through software applications such as Microsoft Word, Excel, Outlook and many others. Most PDAs even allow users to transfer third-party applications such as games or specialized utility programs to their devices.

- **Multimedia Players:** With many users wanting to not only have a device for functionality, they also enjoy being able to use it for entertainment. Having the ability to play music or movies from a PDA eliminates the need for users to carry a second device solely devoted to multimedia playback.

- **Internet connectivity:** Web access and Wi-Fi connectivity allow users to perform tasks as if they were in front of a PC. With built-in applications such as Microsoft Outlook and Internet Explorer, users are just a tap away from email access and web browsing.

- **Bluetooth:** Users are able to wirelessly sync to PCs, transfer files using File Transfer Protocol (FTP) apps, and communicate with other devices such as printers and keyboards using the Bluetooth technology. Bluetooth allows for short range wireless connection between PDAs and peripheral devices that once required cabled connections.

2.1.2 Smart Phones

As with most technological devices, as time progressed, so did device popularity. However, with PDAs not having telephony capabilities, users found themselves carrying two devices: one to organize their lives and one simply for telephony services. PDA users wanted cell phone capabilities with their PDAs and cell phone users wanted PDA functionality with their cell phones. Both of these devices have now converged to create one device, the smart phone. While most specifications do not give a standard or clear cut definition of a smart phone [17] [18] [19], all describe it as essentially a PDA with complete cell phone functionality. Some of the enhanced features of smart phones that usually are not standard on PDAs include:

- **Telephony and Text Messaging:** With the prevalence of cell phone technology in today's society, it only makes sense to add this functionality to PDAs. Adding text messaging and the ability for voice calls has allowed users the option to carry one device, a smart phone, instead of having to keep up with a PDA and cell phone separately.

- **QWERTY Keypads:** QWERTY keypads have the layout of computer keyboards, just in a more compact fashion. They include all alpha-numeric characters and allow users to type alphabetic characters instead of using a number pad to create text. While not popular on most PDAs, QWERTY keypads are seen as an industry standard for smart phones.

- **Global Positioning Systems (GPS):** Newer smart phones allow users GPS capabilities for obtaining exact geographical coordinates and map displays.

Smart phone sales by vendors to end users from 2007 to 2008 were up 13.9% worldwide [20]. While economic conditions haven't exactly been encouraging sales growth lately, IDC, a leading technology research firm, still predicts growth of 3.4% in2009 [21]. With a smart phone having the ability to replace PDAs and cell phones with a single device, one would expect smart phones to soon dominate the market place. For the rest of this document, PDAs and smart phones will now be referred to collectively as PIDs. The classification of PID allows for discussion without distinguishing between the two devices or mentioning both.

2.2 Confidentiality, Authentication, Message Integrity, and Non-Repudiation

Confidentiality, authentication, message integrity, and non-repudiation are all forms of cryptography used to ensure successful communication of clandestine messages between two parties. These cryptographic properties are used in many Internet related transactions highly susceptible to attack, including: online banking, online shopping, and many other venues where sharing of private information is needed.

There are two forms of encryption commonly used to provide confidentiality: private-key cryptography and public-key cryptography. Section 2.5.1 discusses private-key cryptography. Section 2.5.2 then explores the field of public-key cryptography. Finally, section 2.5.3 investigates authentication, message integrity, and non-repudiation, showing how each can be applied to messages or identities to secure data transactions.

2.2.1 Private-Key Cryptography

Private-key cryptography is the oldest form of encryption, dating back centuries into ancient civilizations [22]. Another term for this for private-key cryptography is symmetric cryptography because it only uses one secret key for both the encryption and decryption process, as shown in Figure 2. In order to fully understand this process, there are six basic terms that need to be defined [22].

- **Plaintext:** This is the human-readable message or information that one wishes to make secretive.
- **Ciphertext:** This is an unreadable message that is output by the encryption process.
- **Encryption Algorithm:** This is the process by which plaintext is converted to ciphertext using various substitutions and transformations. Inputs to the encryption algorithm include the plaintext and secret key; the output is ciphertext.

- **Secret Key:** This is the piece of information that must be kept secret. The secret key is used as input into the encryption algorithm so that each plaintext and secret key combination produces different unreadable output. The transformations and substitutions performed by the encryption algorithm are based dependently upon the key.

- **Decryption Algorithm:** This is the encryption algorithm run in reverse. A decryption algorithm takes the secret key and ciphertext as its input and returns the plaintext as its output.

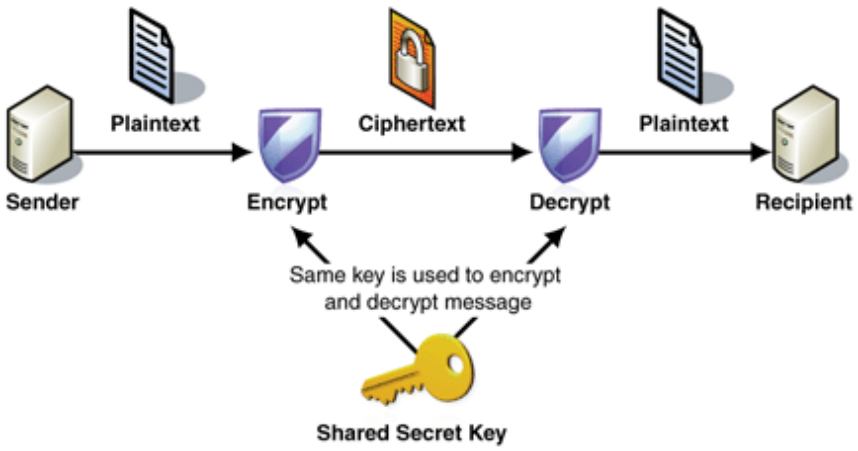


Figure 2. Private-key cryptography [23].

2.2.2 Public-Key Cryptography

Public-key cryptography follows many of the same principles and procedures of private-key cryptography, but is uses different keys in the encryption and decryption process, as shown in Figure 3. For the use of different keys, public-key cryptography is also known as asymmetric cryptography. The keys used in public-key cryptography are defined below [22].

- **Public Key:** This key is public information and is shared so that it can be used by the sender when encrypting a message

- **Private Key:** This key is private information to the recipient and is used in the decryption process.

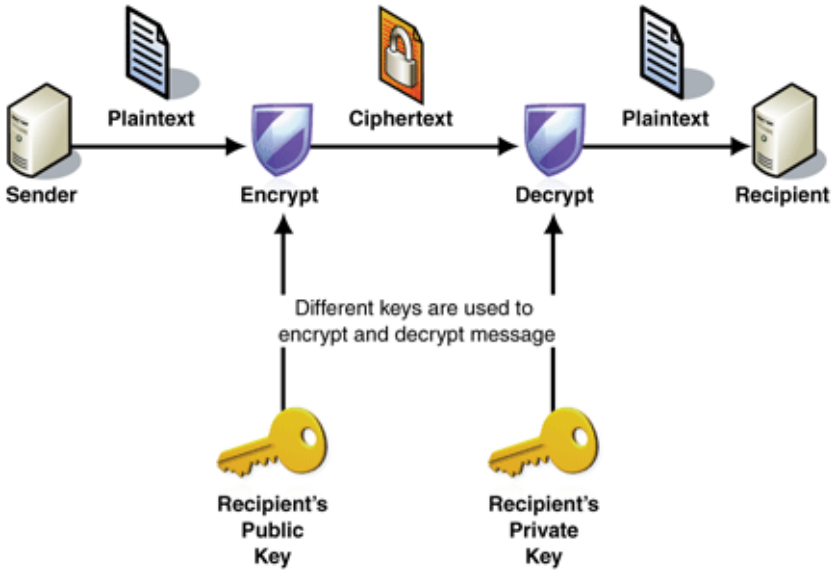


Figure 3. Public-key cryptography [23].

2.2.3 Authentication, Message Integrity, and Non-repudiation

Authentication is the process of verifying that someone is who they claim they are. The ability to prove that data being received is exactly what was sent is known as message integrity. Message integrity verifies that the original message is received with no insertions, deletions, or modifications of any kind. Non-repudiation refers to the receiver's ability to undeniably know that the message being received is from the claimed sender [22].

Although encryption provides data confidentiality, it can also be argued that it provides a degree of authentication, message integrity, and non-repudiation because of the use of shared secret keys. While this might be accepted if it is certain that only two communicating parties truly know the appropriate keys, it becomes completely invalid if a secret key becomes compromised to a third, unintended party. Because of this reason, cryptographers have created two widely accepted cryptographic routines that can be applied to message to successfully achieve the cryptographic properties discussed in this section. The two most common algorithms used to provide authentication, message integrity, or non-repudiation are Message Authentication Codes (MACs) and secure hash functions. Unlike encryption algorithms the process of using a MAC and a secure hash function is one-way, meaning that it is irreversible [22]. Input used to calculate the MAC or hash value cannot be extracted from the resulting output.

- **Message Authentication Codes (MAC):** This algorithm attempts to provide cryptographic security to a message through the use of a fixed length, generated value, known as a MAC. To generate this value, the sender uses a MAC algorithm and two inputs: a shared secret key and the original message. Once the value is generated, it is usually concatenated to the end of the original message and then

sent. Upon receipt, the receiver can generate a MAC, exactly as the sender did, to compare the two values. If the values match, the message has been shown to be cryptographically equivalent to what was sent. If not, it is certain that there was an error in transmission of the message or that the message was altered [22]. The process of using a MAC can be seen in Figure 4.

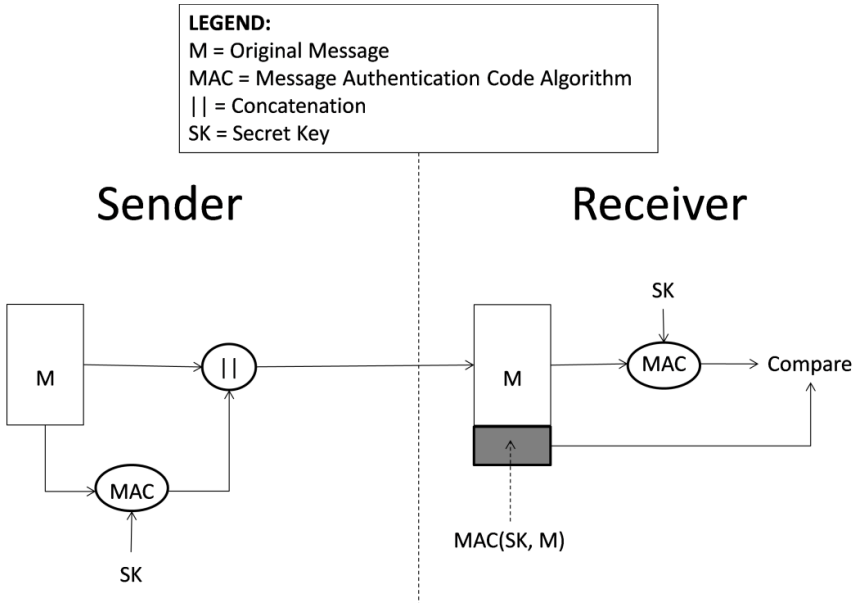


Figure 4. Process of generating a MAC for message authentication, adapted from [22].

- **Secure Hash Functions:** This is an algorithm that attempts to map a variable length message to a fixed length hash value, also known as a message digest. The main difference between secure hash functions and MACs is that hash algorithms only depend upon the message used as input. The only way to check the authenticity of a hash value is for both parties involved to compute the hash value of a message and compare the results [22]. The process of using a secure hash function can be seen in Figure 5.

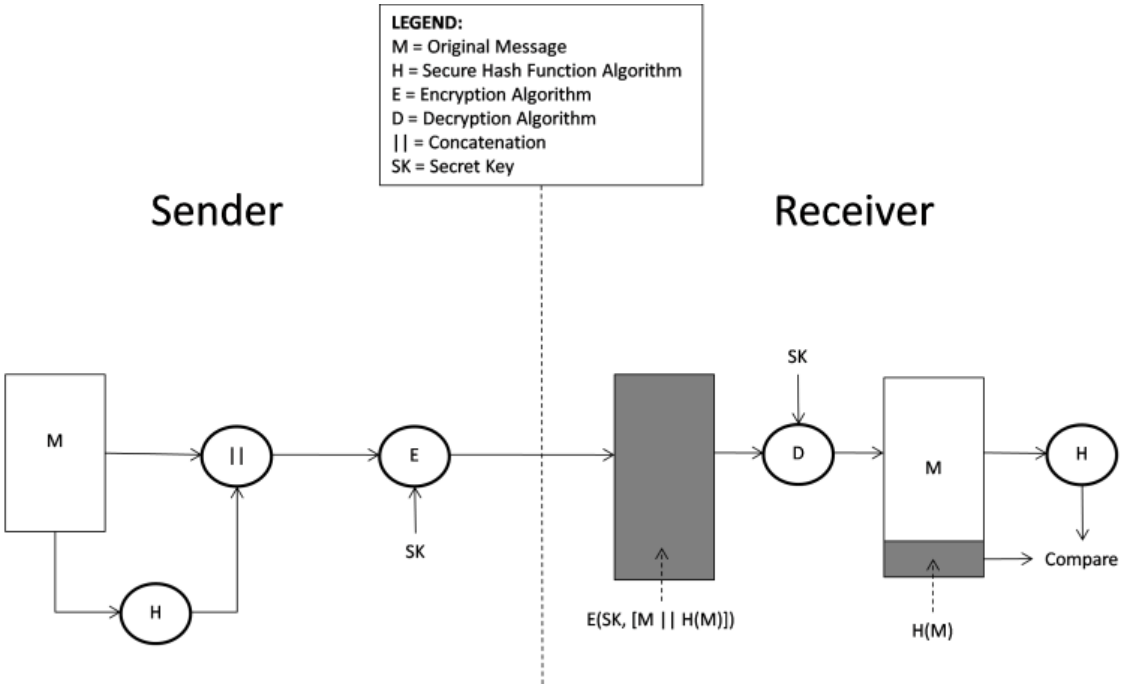


Figure 5. Process of generating a hash value for message authentication, adapted from [22].

2.3 The TCP/IP Suite and Wi-Fi

2.3.1 Layers of the TCP/IP Suite

The Transmission Control Protocol/Internet Protocol (TCP/IP) suite is a 5-layer model, based on the OSI model, used to represent networking protocols and is organized into a hierarchical protocol stack. Each layer is used to encapsulate its own data and provides services to the layer above it. The 5 layers of the stack are: Application Layer, Transport Layer, Network Layer, Data Link Layer, and Physical Layer [24] [25] [26] [27]. The TCP/IP Stack and characteristics can be seen in Table 1.

Table 1. The TCP/IP Model, adapted from [28].

	Layer	Data Unit	Function	Examples
Host Layers	5. Application	Data	Networked applications	HTTP, FTP, POP3, SMTP, SSH
	4. Transport	Segment	End-to-end connections and reliability	TCP, UDP
Media Layers	3. Network	Packet	Path Determination and logical addressing	IP, ICMP
	2. Data Link	Frame	Single hop connections	ARP, NDP
	1. Physical	Bit	Binary transmission	-

- **Application Layer:** At the top of the TCP/IP protocol stack model sits the Application Layer. This layer provides services with which the user is able to interact with such as web browsers, FTP client/servers, SSH clients/servers, and many other applications that are used for network communication.

- **Transport Layer:** This layer is composed of the TCP and User Datagram Protocol (UDP). The purpose of this layer, when using connection-oriented TCP, is to provide end users with reliable data transfers, ensuring that data is delivered to designated places on the receiving computer. In providing reliable data transfers between end users, this layer also has the ability to retransmit segments that are lost in transmission for applications requiring this service. In contrast, UDP offers a connectionless protocol useful for transmissions that must be very fast, such as streaming multimedia, but doesn't provide the reliability, error checking, and resending of packets that TCP implements.

- **Network Layer:** This layer performs routing functions that focus on the end-to-end delivery of packets across a network and includes the IP protocol. Packet fragmentation and reassembly, logical addressing, and network path determination are the key operations performed by devices at this layer. The IP protocol is used extensively by routers trying to deliver data to its next intended hop.

- **Data Link Layer:** When sending data, this layer adds a header and trailer to the network layer data so that hardware can properly recognize starting and ending points of frames transmitted on the physical layer. When receiving data, this layer's primary purpose is to take raw transmissions of bits received and interpret them into Ethernet frames. Once the beginning and end of a frame is recognized, the data link layer strips its frame header and trailer and then passes the remaining frame data on to the network layer so that it is able to perform its intended operations.

- **The Physical Layer:** This layer is the actual physical medium that raw data bits are transmitted through. At this layer, data is converted between digital data and electrical signals so that hardware devices can properly communicate with each other.

When one computer attempts to send data to another computer via the TCP/IP suite, the payload data starts at the top of the sending computer's protocol stack and works its way down to the bottom. At each progressive lower layer, metadata is added to ensure successful delivery of the actual payload data. When the frame transmission reaches the receiving computer, it travels in the reverse direction, or up the protocol stack. Each layer strips off its metadata before passing the actual higher-level data it to the next higher layer. This process can be seen in Figure 6.

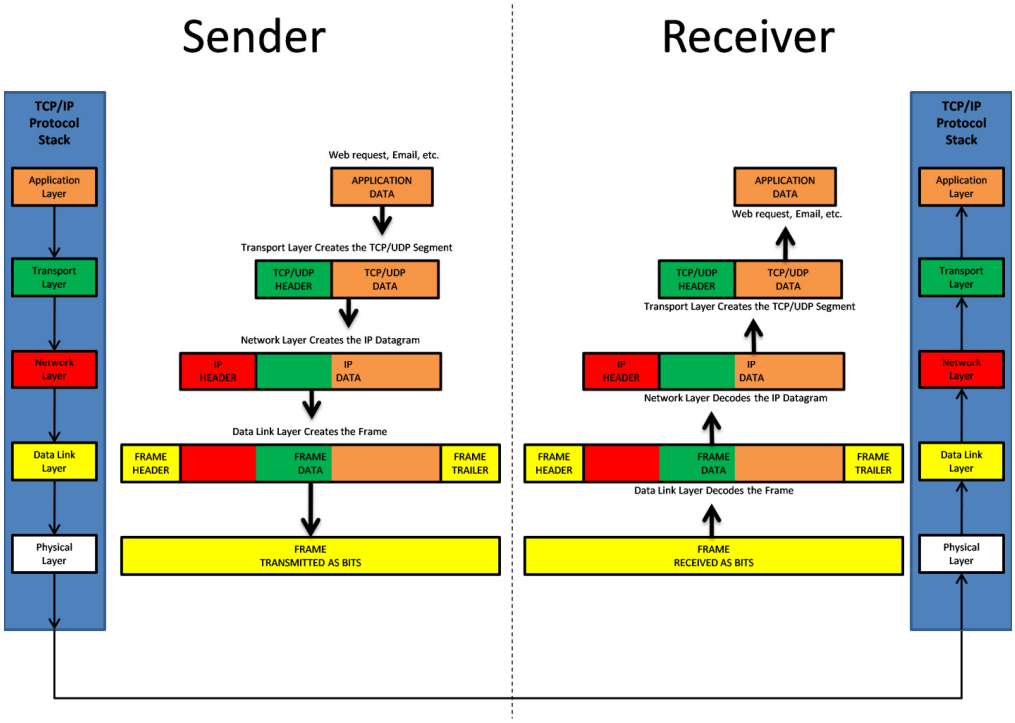


Figure 6. Data communications in the TCP/IP Model, adapted from [24] [25].

2.3.2 Wireless Local Area Networking/Wi-Fi

IEEE 802.11 Wi-Fi is a standard technology for high-speed Wireless Local Area Networking (WLAN) [29]. Wi-Fi uses radio frequencies of 2.4 – 2.5 or 5 GHz in the Industrial, Scientific, and Medical (ISM) band. Wi-Fi functions by creating a WLAN comprised of one or more wireless client nodes each associated with a WAP that connects them to a network at large. In this WLAN, each client node is identified by its Medium Access Control (MAC) address and/or IP address. Wirelessly transmitted 802.11 frames, shown in Figure 8, are used to communicate between wireless client nodes and WAPs [30]. The WLAN/Wi-Fi topology is shown in Figure 7 and a few of the key WLAN/Wi-Fi characteristics are listed in Table 2.

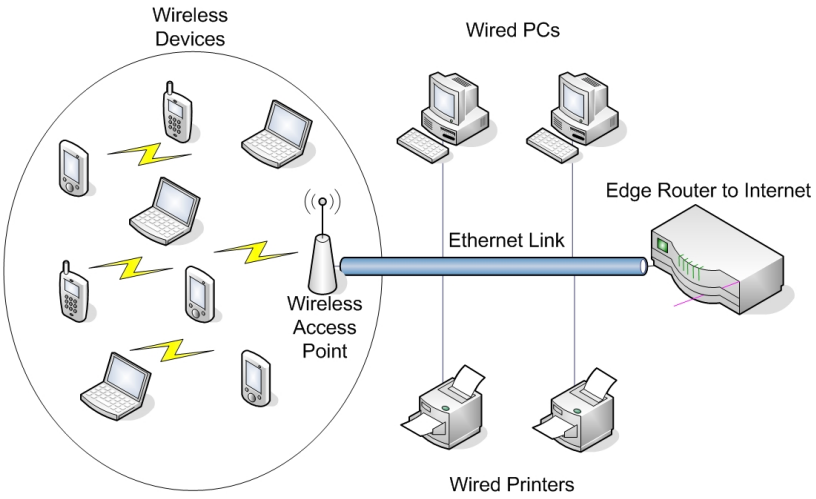


Figure 7. Fundamental 802.11 WLAN/Wi-Fi Topology.

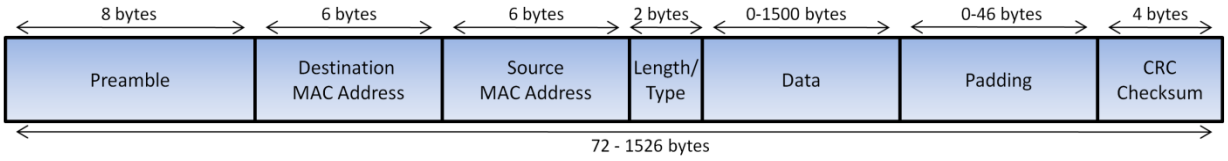


Figure 8. 802.11 Frame Layout.

Table 2. Characteristics of 802.11 WLAN/Wi-Fi Technology [31].

Characteristic	Description
Physical Layer	Direct Sequence Spread Spectrum (DSSS), Frequency Hopping Spread Spectrum (FHSS), Orthogonal Frequency Division Multiplexing (OFDM), Infrared (IR).
Frequency Band	2.4 GHz (ISM band) and 5 GHz.
Data Rates	1 Mbps, 2 Mbps, 5.5 Mbps (11b), 11 Mbps (11b), 54 Mbps (11a), 54 Mbps (11g), 74 Mbps (11n)
Data and Network Security	RC4-based stream encryption algorithm for confidentiality, authentication, and integrity. Limited key management. (AES is being considered for 802.11i.)
Operating Range	Up to 150 feet indoors and 1500 feet outdoors, depending on operating environment.
Positive Aspects	Ethernet speeds without wires; many different products from many different companies. Wireless client cards and access point costs are decreasing.
Negative Aspects	Poor security in native mode; throughput decrease with distance and load.

2.4 The Bluetooth Protocol

In 1998, the Bluetooth Special Interest Group (SIG) was founded to oversee the standardization and perpetuation of the IEEE 802.15.1 Bluetooth protocol. Bluetooth is a wireless technology implementing the idea of a Personal Area Network (PAN). It is designed to be a low cost, low power, short-range wireless solution for the transmission voice and data to/from peripheral devices that once required cabled connections. Many devices using Bluetooth include: cell phones, PDAs, smart phones, hands-free headsets, GPSs, keyboards, printers, etc. With the ease of use that Bluetooth has given its users, it is becoming ever more popular.

To better understand the security ramifications of using Bluetooth enabled devices, one must first understand the underlying architecture and specification. To aid in this, section 2.4.1 provides an overview of the Bluetooth architecture. Section 2.4.2 discusses Bluetooth protocol stack and its organization. Section 2.4.3 explores device discovery. Section 2.4.4 discusses pairing of devices as well as encryption and authentication procedures. Section 2.4.5 investigates the underlying security features built into the Bluetooth protocol and section 2.4.6 discusses Bluetooth security flaws.

2.4.1 Bluetooth Architecture Overview

Bluetooth also operates wirelessly in the 2.4 GHz – 2.485 GHz, ISM band. To avoid interference with other wireless devices transmitting in the ISM band and to mitigate eavesdropping, Bluetooth employs two frequency hopping schemes. Adaptive frequency hopping avoids interference with other wireless devices by detecting and avoiding frequencies being used by those devices. This proves useful if Bluetooth devices are communicating in the same airspace as Wi-Fi technology that transmits on fixed frequencies in the 2.4 GHz, ISM band. To mitigate eavesdropping, spread spectrum frequency hopping allows devices to transmit packets on 79 different RF channels, each 1 MHz apart, at a rate of up 1600 hops/second. Each RF channel is divided into time slots, each 625 µs long with master devices transmitting on even time slots and slave devices transmitting on odd time slots. Master devices are classified as those that initiate a connection and slaves are the partner device of the linked connection. Devices send packets in these time slots and one packet can use up to five consecutive time slots if circumstances permit [31] [30] [32] [33] [34] [35]. The basic Bluetooth packet format is shown in Figure 9.

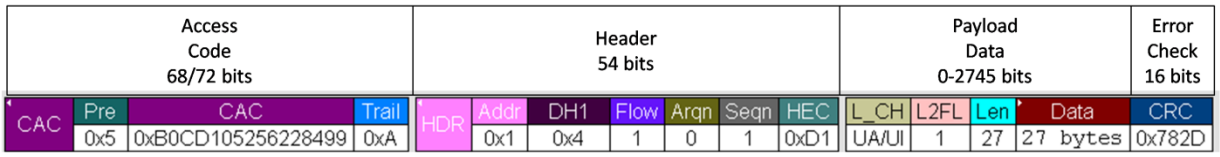


Figure 9. Basic Bluetooth packet format.

Bluetooth enabled devices pair with other Bluetooth-enabled devices to create a piconet. Piconets are simply small groups of paired Bluetooth devices that form a PAN. In a piconet, there is a single master device with up to seven actively communicating slave devices paired with it. The number of active slaves is limited to 7 because of the 3-bit Active Member Address assigned to each slave device. Devices that are not actively communicating, but still paired with the master device are said to be in a parked state. A piconet can support up to 256 parked slaves using a 7-bit Parked Member Address. A single device can be part of more than one piconet, either being a slave in multiple piconets or a master device in one and slave device in another. This piconet interconnection forms a scatternet. In a single scatternet, up to 10 piconets can be interconnected before performance degradation is noticeable [31] [30] [32] [33] [34] [35]. This paired communication topology can be seen in Figure 10.

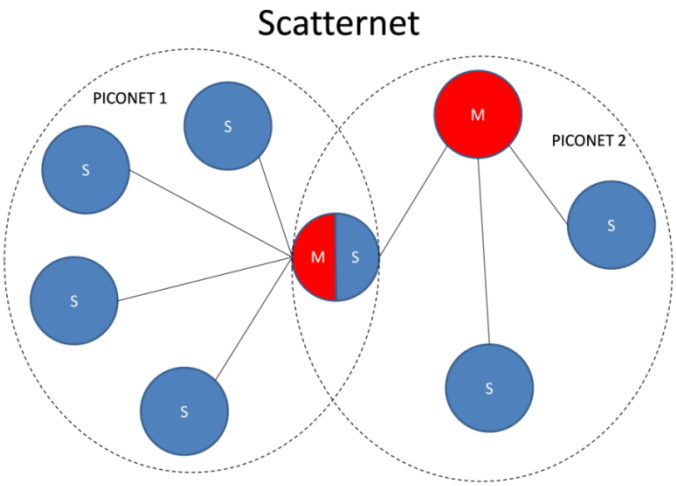


Figure 10. Bluetooth piconet/scatternet topology.

When two Bluetooth enabled devices pair, the links between the master and slave can be of three different types [31] [30] [32] [33] [34] [35]:

Synchronous Connection Oriented (SCO):

- Used for voice communications
- Implemented using circuit switching (Phone Switching)
- Provides synchronous and symmetrical services
- Slot reservation at fixed intervals

Extended SCO (ESCO):

- Asymmetric links
- Supports greater number of packet types

Asynchronous Connection-less:

- Used for data communications and signaling
- Implemented using packet switching
- Supports symmetrical and asymmetrical asynchronous services
- Used for device discovery and paging

2.4.2 Bluetooth Protocol Stack

The Bluetooth protocol stack, shown in Figure 11, is arranged into a tiered hierarchy much like that of the TCP/IP model. Each lower layer in the stack provides data and services to the layer directly above it.

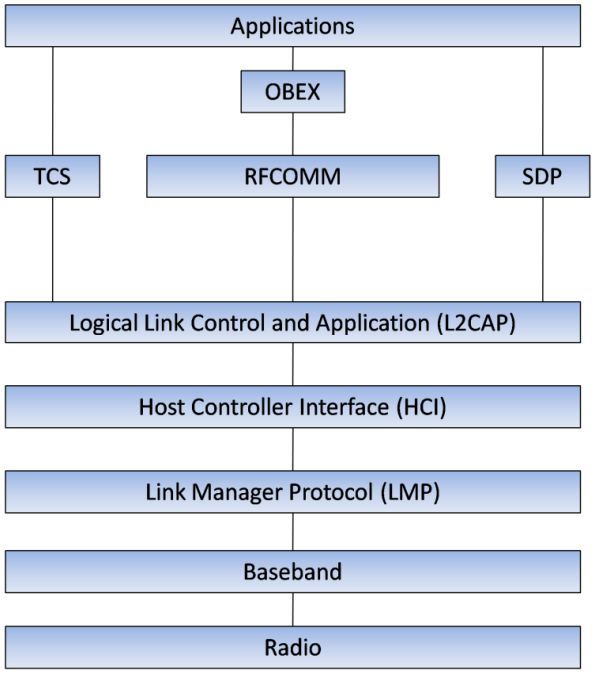


Figure 11. Bluetooth protocol stack.

The Radio Layer provides the actual modulation and demodulation of digital data to RF waves using Gaussian Frequency Shift Keying (GFSK). The Baseband Layer controls physical links between devices using the Radio Layer, assembles packets, and controls frequency hopping. The Baseband Layer also controls the operation of the Radio layer by allowing it to operate in one of three different classes, each with different transmission ranges and power ratings. Most devices operate in PAN environments (Class 2), which only demand limited range and minimal power consumption [31] [30] [32] [33] [34] [35]. The three allowable radio classes are shown in Table 3.

Table 3. Bluetooth transmission classes.

CLASS	RANGE (meters)	POWER (mW)
1	100	100
2	10	2.5
3	1	1

The Link Manager Protocol (LMP) configures and controls all links to other devices. The Host Controller Interface (HCI) acts as a liaison between upper application layers and lower hardware layers. It has the ability to change configuration parameters of the Baseband and LMP layers through standardized commands. The Logical Link Control and Adaptation (L2CAP) Layer allows for segmentation and reassembly of packets , flow control, quality of service through retransmission of unacknowledged packets, and multiplexing of higher layer protocols to lower layer links. The Baseband Layer limits the size of single packets for transmission purposes, but the L2CAP Layer allows for the transfer of larger packets through the use of segmentation and reassembly, shown in Figure 12 [31] [30] [32] [33] [34] [35].

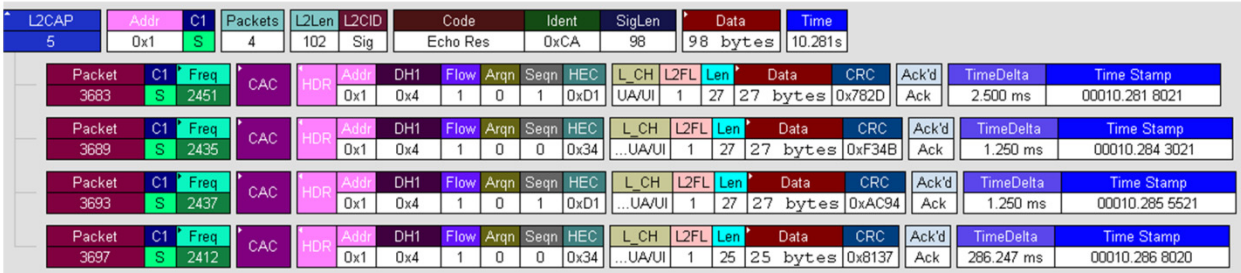


Figure 12. L2CAP Layer packet reassembly.

After the L2CAP layer, the protocol stack becomes more of a tree-like structure, allowing for protocols based on user needs. The Radio Frequency Communication (RFCOMM) protocol emulates RS-232 functionality, but through a wireless means. The Service Discovery Protocol (SDP) allows Bluetooth devices to discover what services are offered by other devices and the Telephony Control Protocol Specification (TCS) provides telephony services. Above the RFCOMM layer is another liaison layer known as the Object Exchange (OBEX) layer. The OBEX layer simply allows for the transmission of data objects between devices. Above all of these protocols is the Application Layer. This layer is used for Bluetooth profiles that give specifications for how applications can utilize the Bluetooth protocol stack [31] [30] [32] [33] [34] [35].

2.4.3 Device Discovery

For device discovery, the Bluetooth device address is the essential piece of information. Connections to other devices are impossible without it. A Bluetooth device address is divided down into three components: the Lower Address Part (LAP), Upper Address Part (UAP), and Non-significant Part (NAP) [36]. The LAP is 24-bits in length and model/device specific. The LAP can be found in packet headers and validated using the checksum calculated in the Cyclic Redundancy Check (CRC) field of the packet. The UAP is 8-bits in length and only transmitted in the Header Error Code (HEC) field of packets negotiating the pairing of devices. The HEC contains the result of a linear feedback algorithm that is initialized with the UAP. By reversing the algorithm at the receiving

end, the UAP can be determined. The NAP is 16-bits in length, with the most significant 8-bits almost always zeros. Although the NAP is used to identify a device, it has no impact on the generation of packets and is not contained in them. The NAP and UAP together is vendor specific and known as the Organizationally Unique Identifier (OUI). Bluetooth device addresses are all unique and exclusive address ranges are assigned to individual vendors by the IEEE [37]. This is done to divide the Bluetooth address space and to circumvent duplicate addressing. Figure 13 shows a Bluetooth device address and a breakdown of its components.

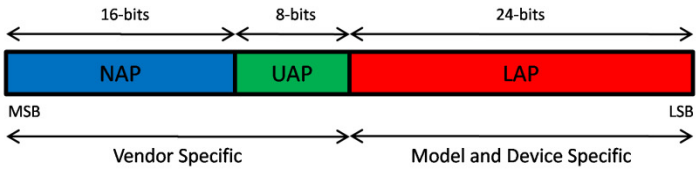


Figure 13. The components of a Bluetooth device address.

There are two ways to obtain a Bluetooth device address. If the device address is known, a connection attempt to the address can result in successful pairing. If the device address is unknown, the process is a bit more complex. When a device wants to identify all devices in its vicinity it sends a broadcast inquiry request that other Bluetooth devices can reply to. However, devices only reply if they are in the appropriate mode of operation. The three Bluetooth operation modes are [31] [30] [32] [33] [34] [35]:

- **Off:** The Bluetooth radio is disabled and no Bluetooth communication is possible.
- **Non-Discoverable Mode:** Bluetooth devices receive the inquiry request, but discard it. This is said by most to be a security feature because if the device does not respond to inquiry requests, in effect, only devices that know its Bluetooth device address can connect to it. This essentially allows Bluetooth devices to operate in a stealth mode invisible to others.
- **Discoverable Mode:** When a device is in this mode, it responds to all inquiry requests, allowing other devices to know it is nearby and able to connect.

When the Bluetooth device address is successfully obtained, the two devices can then be paired for secure communication purposes.

2.4.4 Device Pairing, Authentication, and Encryption

2.4.4.1 Bluetooth Key Generation

When two devices need to communicate in a secure manner, they must go through an authentication process and establish an encryption key. To do this, two keys must be generated: a link key (also known as unit key) and an encryption key. A mutual and secret Personal Identification Number (PIN) must be used by both parties to generate these keys. Bluetooth PINs are 0-16 case-sensitive, alphanumeric characters, but many users only use 4 decimal digits so that the PIN is easily remembered [33]. The process of generating the link key and encryption key through the use of the SAFER+ encryption algorithm is shown in Figure 14 [31] [35].

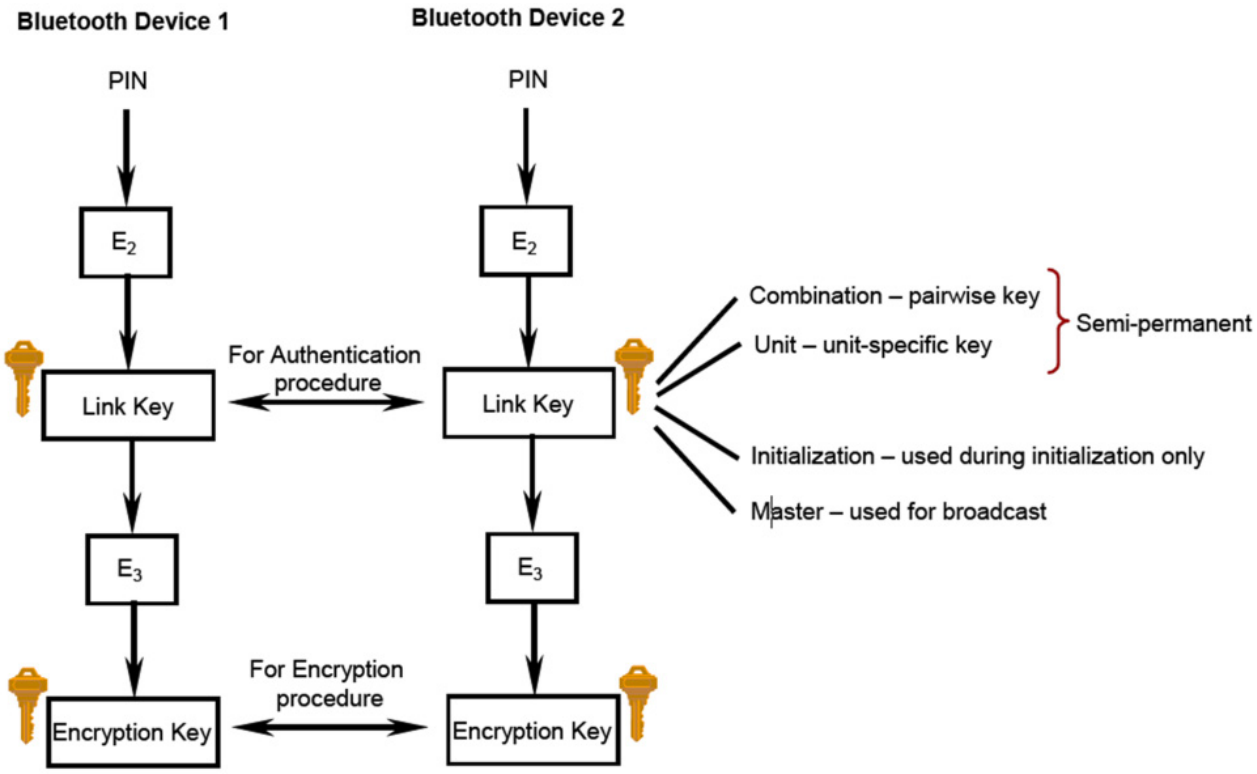


Figure 14. Bluetooth key generation from PIN [31].

2.4.4.2 Bluetooth Authentication Prior to Core Specification v2.1

Prior to the Bluetooth Core Specification v2.1 + Enhanced Data Rate (EDR), mutual authentication of two connecting devices followed the process shown in Figure 15. Even though the process is outdated by the newest specifications, it is relevant because many Bluetooth devices operating today were deployed prior to the newest specification update [31] [35].

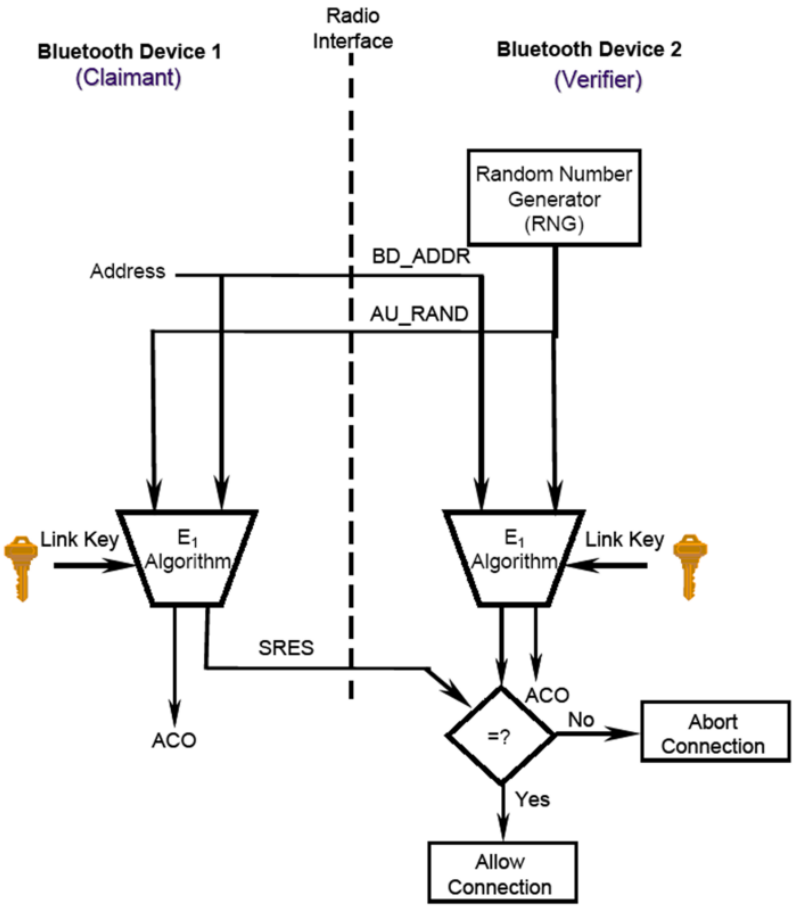


Figure 15. Bluetooth authentication process [31].

2.4.4.3 Bluetooth Authentication Using Core Specification v2.1

With the Bluetooth SIG recognizing the ability to perform passive eavesdropping attacks on the authentication process prior to the v2.1 + EDR update, they introduced a new authentication process known as secure simple pairing. This method, shown in Figure 16, utilizes Elliptic Curve Diffie-Hellman (ECDH) public-key cryptography [31] [35].

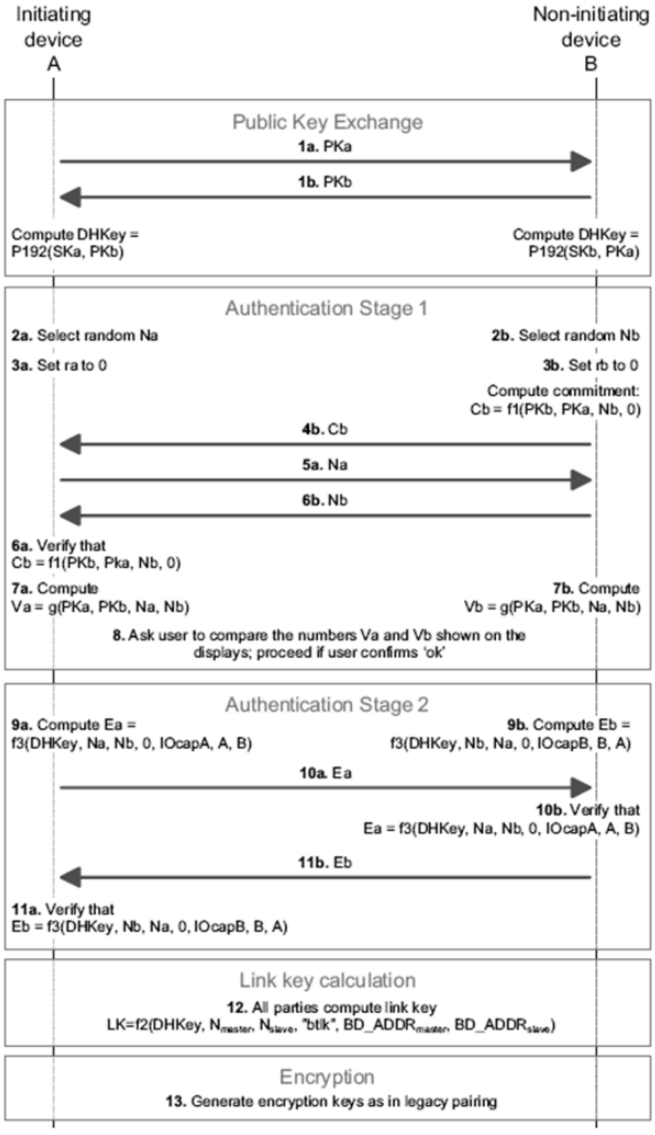


Figure 16. Bluetooth authentication using Secure Simple Pairing [38].

2.4.4.4 Bluetooth Encryption Process

Once two Bluetooth enabled devices are authenticated, they are then able to communicate using encryption based on the mutually generated encryption key. The process of encrypting Bluetooth data for transmission is shown in Figure 17 [31] [35].

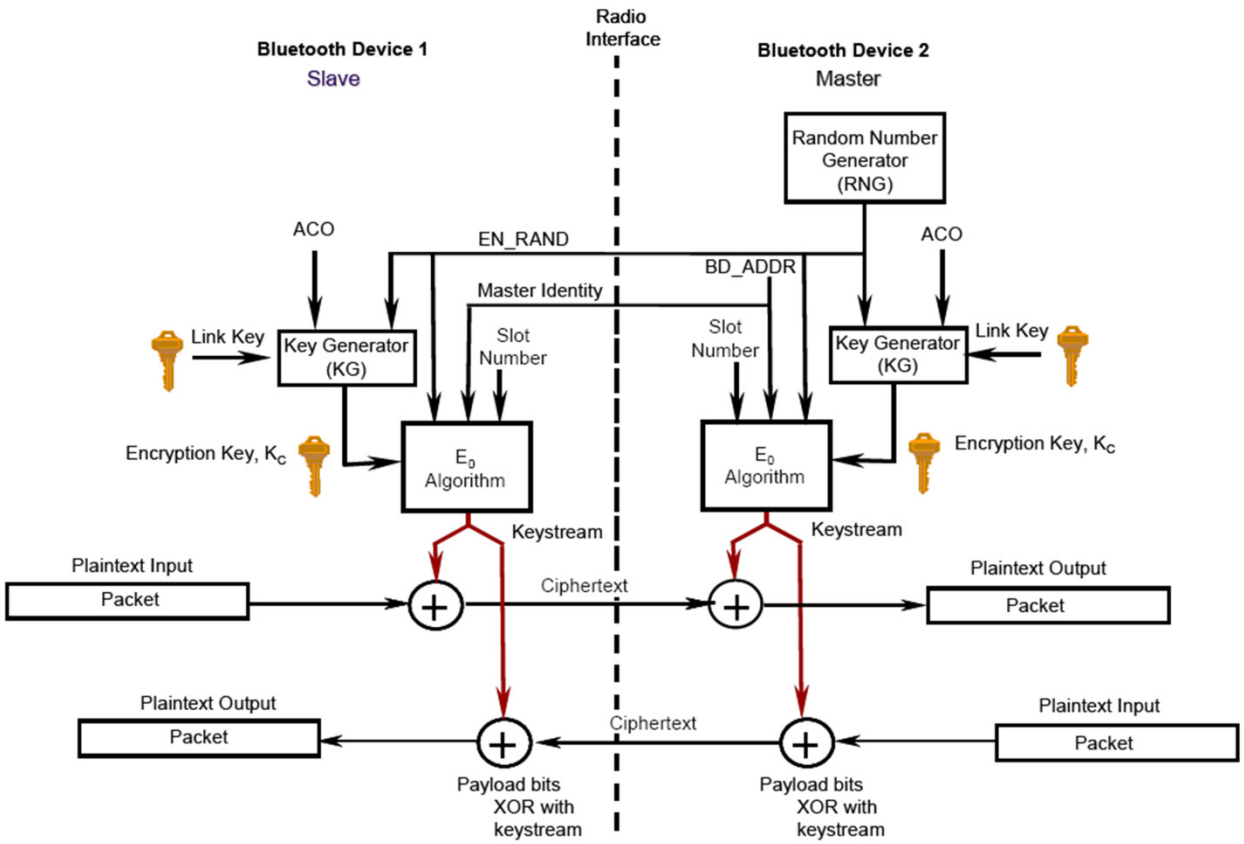


Figure 17. Bluetooth encryption process [31].

2.4.5 Bluetooth Security Features

2.4.5.1 Security Services

When the Bluetooth SIG designed the protocol specification, they did so with the need for security in mind. The security services implemented by the specification are [31] [35]:

- **Authentication:** This security feature tries to verify the identity of the two devices attempting to communicate. Without it, devices can't be sure who they are communicating with.
- **Confidentiality:** This security feature attempts to provide privacy to communicating devices through the means of encryption. By doing so, it attempts to thwart eavesdropping attacks in which attackers passively monitor traffic.
- **Authorization:** This security feature allows for the control of resource access and usage.

2.4.5.2 Security Modes

The Bluetooth specification allows Bluetooth devices to operate in one of four different security modes [31] [35].

- **Security Mode 1:** This mode is also known as the unsecure mode. It provides no authentication or encryption during Bluetooth transmissions.
- **Security Mode 2:** This mode is also known as the service level security mode. It provides authentication and encryption at the L2CAP layer, meaning that security is enforced between devices after the connection link has already been established. This mode introduces the security service of authorization, allowing controlled access to services on the device.
- **Security Mode 3:** This mode is also known as the link level security mode. It provides authentication and encryption at the Baseband layer, meaning that security procedures take place prior to any device connection.
- **Security Mode 4:** This mode was introduced when Bluetooth v2.1 + EDR was released. It was simply created for the use of Secure Simple Pairing, which greatly enhances the mutual authentication process of Bluetooth communication.

2.4.5.3 Encryption Modes

The Bluetooth specification also allows for three different modes of encryption to support the confidentiality service [31] [35].

- **Encryption Mode 1:** Encryption is not performed on any Bluetooth traffic.
- **Encryption Mode 2:** All traffic to and from individually specified devices is encrypted. Broadcast traffic is not protected in this mode, it is sent unencrypted.
- **Encryption Mode 3:** All traffic is encrypted.

2.4.5.4 Trust and Service Levels

In addition to the security modes and encryption modes, the Bluetooth specification also allows for two different levels of trust. Trusted devices are those that maintain a permanent relationship with another device, thus giving them full access, or authorization, to all services. Untrusted devices are those that do not maintain a permanent relationship with another device, meaning that they have restricted access to services on another device.

For services, there are also different levels of security. Service security is also another mode of Bluetooth security that offers three levels of support [31] [35].

- **Service Level 1:** This level requires authorization and authentication. Trusted devices are automatically granted access, but untrusted devices need to be manually authorized.
- **Service Level 2:** This level requires only authentication, authorization is not needed. Access to services is grated after the authentication process has been completed.
- **Service Level 3:** Authentication and authorization are not required at this service level. Access is automatically granted to the connecting device.

2.4.6 Current Bluetooth Security Flaws

Although the Bluetooth protocol was designed to be as secure as possible, like all protocols, it has its weaknesses. While not all security flaws reside in the Bluetooth protocol specification, many are due to poor implementations by manufactures or software implementations that don't completely following the specification.

Table 4 discusses a few of the existing Bluetooth security issues and reasons why they can be used to exploit vulnerabilities in Bluetooth-enabled devices [31] [35].

Table 4. Current Bluetooth security weaknesses [35].

	Security Issue or Vulnerability	Remarks
Versions Before Bluetooth v1.2		
1	Unit key is reusable and becomes public once used.	A unit key should be used as input to generate a random key. A key set should be used instead of only one unit key.
2	Unit key sharing can lead to eavesdropping.	A corrupt user may be able to compromise the security between two other users if the corrupt user has communicated with either of the other two users. This is because the link key (unit key), derived from shared information, has been disclosed.
Versions Before Bluetooth v2.1		
3	Short PINs are allowed.	Weak PINs, which are used for the generation of link and encryption keys, can be easily guessed. People have a tendency to select short PINs.
4	PIN management is lacking.	Establishing use of adequate PINs in an enterprise setting with many users may be difficult. Scalability problems frequently yield security problems.
5	Encryption key stream repeats after 23.3 hours of use.	The encryption key stream is dependent on the link key, a random number generator, the master Bluetooth device address, and clock. Only the master's clock will change during a particular encrypted connection. If a connection lasts more than 23.3 hours, the clock value will begin to repeat, hence generating an identical key stream to that used earlier in the connection.
All Versions		
6	Link keys are stored improperly.	Link keys can be read or modified by an attacker if they are not securely stored and protected via access controls.
7	Attempts for authentication are repeated.	A limiting feature needs to be incorporated in the specification to prevent unlimited requests. The Bluetooth specification currently requires a time-out period between repeated attempts that will increase exponentially.
8	Strength of the challenge-response pseudo-random generator is not known.	The Random Number Generator (RNG) may produce static number or periodic numbers that may reduce the effectiveness of the authentication scheme.

9	Encryption key length is negotiable.	The specification allows devices to negotiate encryption keys as small as one byte. A more robust encryption key generation procedure needs to be incorporated in the specification.
10	The master key is shared.	A better broadcast keying scheme needs to be incorporated into the specification.
11	No user authentication exists.	Only device authentication is provided by the specification. Application-level security, including user authentication, can be added via overplay by the application developer.
12	The E_0 stream cipher algorithm used for Bluetooth encryption is weak.	More robust encryption needs to be incorporated in the specification.
13	Privacy may be compromised if the Bluetooth device address is captured and associated with a particular user.	Once the Bluetooth device address is associated with a particular user, that user's activities could be logged, resulting in a loss of privacy.
14	Device authentication is simple shared-key challenge-response.	One-way-only challenge-response authentication is subject to man-in-the-middle attacks. Bluetooth provides for mutual authentication, which should be used to provide verification that users and the network are legitimate.
15	End-to-end security is not performed.	Only individual links are encrypted and authenticated. Data is decrypted at intermediate points. End-to-end security on top of the Bluetooth stack can be provided by use of additional security controls.
16	Security services are limited.	Audit, nonrepudiation, and other services do not exist. If needed, these services can be incorporated in an overlay fashion by the application developer.
17	Discoverable and/or connectable devices are prone to attack.	Any device that must go into discoverable or connectable mode to pair should only do so for a minimal amount of time. A device should never be in discoverable or connectable mode all the time.

2.5 Intrusion Detection Systems

According to Sundaram [39], an intrusion attempt is the possibility of a deliberate unauthorized attempt to access information, manipulate information, or render a system unreliable or unusable. The need to recognize and respond to these intrusion attempts has lead to the development of IDSs. An IDS is a security system that monitors computer systems along with network traffic to detect possible attacks [40].

IDSs can be deployed in two ways, either monitoring an individual system or an entire network. Single system monitoring is said to be host-based. Monitoring of entire networks at their borders, protecting multiple systems, is known as a network-based IDS. IDSs are compromised of two main components:

1. **Sensors:** These devices monitor network traffic and record malicious activities as security events.

2. **Management Consoles:** These devices allow security administrators (SAs) to control sensors and view alerts. A management console can also be used to log recorded events to a centralized database for later forensic analysis.

IDSs can monitor computer networks in a variety of ways to recognize malicious packet streams. They can be categorized into three main types based on the approach used to detect network attacks: signature-based, anomaly-based, and hybrid systems.

2.5.1 Signature-Based IDSs

A signature-based IDS is built based on the idea that attacks can be recognized primarily on a flow of packets that have predictable patterns, also known as signatures. Systems that employ signature-based IDSs are used extensively in industry because they often yield very accurate and specific detection results. The disadvantage to using signature-based IDSs is that only known attacks in the signature database will be used when analyzing network traffic. If a new attack is developed, it will be unrecognizable to a signature-based IDS. However, an advantage to using a signature-based system is that as soon as a new attack is known and understood, an attack signature can be added to the IDS's attack signature database to always recognize the attack in the future.

2.5.2 Anomaly-Based IDSs

IDSs of this type are built on the premise that all attacks are anomalous in nature, meaning that they do not follow the predicted pattern of network traffic norms. These systems can be built using two different approaches.

- **Predictive Pattern Recognition:** This method tries to predict future events based on those that have already occurred. This system also uses signatures, or rules, but not in the same way as traditional signature-based IDSs. Rules for this type of system are based on the idea that event 1 and 2 have taken place and thus predicting that there is an X% chance of event 3, Y% chance of event 4, and a Z% chance of event 5 following, etc. If the network flow matches the predictive pattern rule, the IDS can flag the packet stream as intrusive and generate an alert. A disadvantage to this method is that event 6 could follow events 1 and 2, but since event 6 is not listed in the rule, the network stream will not be flagged as intrusive. Advantages of these systems are that they detect sequential patterns that were once hard to recognize and they also are highly adaptive to changes much like the signature-based IDSs [39].

- **Statistical Analysis:** To use this method, behavior profiles are first generated to accurately depict "normal", or baseline, system behavior. Several factors are used to build a behavior profile, including: CPU time used, network connections in a time period, and other activity measures. Once a system has a behavior profile it is then able to monitor network traffic for behavior that deviates from baseline profile. The advantage of using statistical analysis is that over time, the IDS can learn the behavior of its users. However, this also allows for a serious disadvantage. Gradually, over time, attackers can train the system to consider intrusive events or behavior as normal activity [39].

Even with the considered advantages to using anomaly-based IDSs, there are two main issues that make anomaly-based IDSs less desirable for industry use. First, they are only effective when correct thresholds have been set as baseline system normalcy, which in most cases, is difficult to do. The second drawback is that due to the overhead of tracking network streams and storing state information, these systems are very computationally and resource expensive.

2.5.3 Hybrid IDSs

A hybrid IDS simply employs using both a signature-based module and an anomaly-based module to recognize attacks. In combining the two approaches, a hybrid IDS can utilize the benefits of both methods, while mitigating the drawbacks of each.

3. Related Work

Implementing a hybrid intrusion detection system is by no means a new or ground-breaking idea, but doing so for PIDS using power anomaly triggers coupled with real-time Wi-Fi and Bluetooth attack correlation is a field that has not been explored until now. While this approach is innovative in many ways, it would not be possible without many prior research efforts. Section 3.1 discusses previous work pertaining to power conservation in order to extend mobile device lifetimes. Section 3.2 explores an approach to attacking mobile devices through targeting their limited power resources as an exploitable vulnerability. Section 3.3 investigates prior research that has focused on power anomaly intrusion detection in PIDs. Section 3.4 introduces Snort and its use as an IDS. Section 3.5 discusses previous work that has been done in the field of Bluetooth intrusion detection.

3.1 Extending Mobile Device Life

Power conservation in mobile devices is of paramount concern. If device life can be prolonged, users can be more productive and more satisfied with the use of the device. Section 3.1.1 explores the field of smart batteries and how they can extend mobile device time and increase performance. Section 3.1.2 introduces two of the advanced power conservation standards that have been developed to aid in the effort for more power conscious devices.

3.1.1 Smart Batteries

An expectation of prolonged battery life has led to the development of the Smart Battery System (SBS) [41]. SBS is a system used to control, monitor and conserve battery power in mobile devices ranging from PIDs to mobile medical equipment. A smart battery utilizes embedded electronics to store smart battery data (voltage, current, remaining capacity, run-time-to-empty, etc...) and operating parameters, which in turn allows the SBS to predict and optimize battery performance for extended mobile device run-times [41].

3.1.2 Advanced Power Conservation Standards

Advanced Power Management (APM) [42] and Advanced Configuration and Power Interface (ACPI) [43] have been created for the purpose of standardizing power conservation techniques through the use of industry-common configuration interfaces. APM is a Basic Input Output System (BIOS) - based layered software standard

that allows higher-level software to interact with operating systems and device drivers to reduce power consumption without the need of knowing the hardware interfaces [42]. The main idea behind APM is to control power usage of a system based on system activity, meaning if system activity decreases, so does the power to system resources.

ACPI is an industry specification that builds upon the older APM standard to further enhance programming interfaces for power management purposes. The purpose of this specification is to create an industry-wide standard for the configuration of motherboard power management.

Power management can be enhanced and standardized on an industry-wide scale with the creation of the APM and ACPI specifications. This not only simplifies the realm of power management in computer systems, but also improves performance and allows for longer operating lifetimes of those devices using battery-powered hardware.

3.2 Attacking Battery-Powered Mobile Devices

Even with the SBS, APM, ACPI, and other power conservation techniques, attackers are exploiting mobile devices through DoS attacks targeting rapid battery depletion. These attacks are known as sleep deprivation torture, or battery exhaustion attacks [11] [12]. When mobile devices are inactive, or not in high need of system resources, they enter sleep mode. This allows the device to enter a state of minimal power consumption and process suspension. If an attacker can keep a mobile device in a high rate of power consumption without allowing it to sleep, the device will become inoperable much faster than expected due to insufficient battery resources. It becomes difficult to detect and defend against with this being a novel approach for a DoS attack.

Martin et al. [12] further investigated this approach of attacking mobile devices and introduced a way to classify different variations of these attacks. Based on their classification system, battery exhaustion attacks could be implemented in three different ways:

- **Service Request Power Attacks:** This category of attack targets battery depletion by attackers making repeated request to victim devices for certain services. These services are usually network-based, aimed at draining the battery through increased Wi-Fi communication on the wireless Network Interface Card (NIC).

- **Benign Power Attacks:** An attack of this nature forces victim devices to repeatedly perform tasks that consume vast amounts of battery power. This form of attack is hidden to the user, but it is not something that is intended to harm the device in any way other than to accelerate the process of

battery depletion. Requiring a mobile device to execute hidden Java script is an example of a benign power attack.

- **Malignant Power Attacks:** This type of attack is not only aimed at draining the battery, but also being harmful to the overall operation of the device. Attacks of this form can make changes to the operating system kernel or change application binaries so that more power is drained during execution. These attacks are usually implemented in the form of viruses or Trojan horses and target increasing CPU clock frequency.

3.3 Identifying and Preventing Battery Exhaustion Attacks

Researchers have begun to put forth effort to try to mitigate the effectiveness of battery exhaustion DoS attacks on mobile devices. Section 3.3.1 investigates an approach aimed at detecting battery exhaustion attacks on laptop PCs. Section 3.3.2 discusses the Battery-Based Intrusion Detection System that pioneered the field of using power anomaly as an intrusion detection mechanism for PIDs. Section 3.3.3 introduces the Battery-Sensing Intrusion Protection System which formed a foundation for this research and the creation of MVP-IDS.

3.3.1 Detecting Battery Exhaustion on Laptop Devices

Nash et al. [44] observed a need for detecting battery exhaustion attacks and produced a viable prototype for laptop computers that could accomplish this on a per process basis. This approach used system performance parameters, such as, CPU load, disk reads/writes, and network transmissions to first estimate correlation coefficients using a multiple linear regression model. By doing so, the coefficients could then be used to model and estimate power usage of the system as a whole. Battery exhaustion can be detected when power expenditures exceeded the estimation for extended time periods using the power estimation model. Another feature of the IDS is the ability to map power consumption on a per process basis. Each process is monitored to allow for the detection of processes consuming large amounts of processor usage. Since processor usage is the largest factor in power consumption, a process aimed at battery exhaustion would have a higher processor usage than that of most other processes on the system.

3.3.2 Battery-Based Intrusion Detection System (B-BID)

Jacoby also attempted to solve the problem of battery exhaustion attacks by creating the Battery-Based Intrusion Detection System (B-BID) [45] [46] [47]. This was the first power anomaly-based IDS intended for securing PIDs. B-BID incorporates three modules to monitor power consumption and to correlate anomalies with network based connections.

- **Host Intrusion Detection Engine (HIDE):** This module is a rules-based engine that attempts to detect power expenditure abnormalities based on device power consumption characteristics and static thresholds based on PID power states.

- **Scan Port Intrusion Engine (SPIE):** This module operates similar to netstat on a desktop or laptop PC. Netstat is a command-line application is used to monitor incoming/outgoing network connections, network interface statistics, and routing tables. Once an attack is detected by HIDE, SPIE logs running processes and network connection information including: timestamp, source IP address, destination IP address, source port and destination port. This data allows for forensic analysis for the discovery of attack sources.

- **Host Analysis Signature Trace Engine (HASTE):** This module attempts to identify attacks based on energy expenditure signatures created using a Fast Fourier Transform (FFT). HASTE uses a process that "distinguishes the dominant frequency (x) from amplitude (y) peaks, consistently producing unique xy plots that effectively differentiate attacks. [47]"

3.3.3 Battery-Sensing Intrusion Protection System (B-SIPS)

Jacoby was able to produce a viable solution for securing PIDs against battery exhaustion attacks with the invention of B-BID. Directly from his research evolved the Battery-Sensing Intrusion Protection System (B-SIPS) [1]. B-SIPS is a client/server-based model used to also detect power anomalies in PIDs developed by Buennemeyer et al. This research extended Jacoby's work by introducing the DTC algorithm [1] which attempted to mitigate false positive intrusion alerts by analyzing power consumption characteristics and recalculating device power expenditure thresholds every second. Many of the original B-SIPS features [1] are still included in MVP-IDS; those that are will be discussed in later sections of this document.

3.4 Snort

Network-based attacks, such as viruses, worms, Trojan horses, and buffer overflows, have now become ubiquitous as the number of network-based users has grown. Snort [2], a widely used and highly regarded IDS, can be used to combat these attacks and protect network assets by monitoring network packet streams for malicious activity. Since Snort is accepted as an industry standard in intrusion detection, with a free open source license, it was chosen to implement the Wi-Fi IDS module for MVP-IDS.

3.5 Bluetooth Intrusion Detection

Bluetooth exploits have emerged as a popular vector for attackers, mainly because of the increasing extent to which the technology is being deployed [48]. O'Connor [49] first developed a network-based Bluetooth IDS to discover mobile devices under attack within Bluetooth piconets. Once attacks are discovered, responses are deployed such as honeypots, false messaging, and target cloning to disrupt or prevent further attack. Because of this IDS and the data it collected, many Bluetooth attacks now have known signatures. Attacks can be detected using a Bluetooth protocol analyzer and a rules engine to signature-match attack traffic. Moreover, because this work produced an incredible breakthrough in Bluetooth intrusion detection and created signatures for common Bluetooth attacks, its signatures were used to develop BADSS, the Bluetooth IDS module for MVP-IDS.

4. MVP-IDS Design and Methodology

This chapter details the MVP-IDS design and methodology. The main objective of this research was to hinder outside sources from negatively influencing the usability and lifetime, per battery charge, of PIDs. The methodology of MVP-IDS was aimed at not only identifying IC anomalies from B-SIPS clients, but also validating those reports with real-time Wi-Fi and Bluetooth attack traffic, thus giving confidence of intrusion detection to the system as a whole. This chapter is broken into four sections, each describing a module or significant component of the MVP-IDS design. Section 4.1 details the use of B-SIPS clients for IC anomaly triggering and enhancements made to the previous version. Section 4.2 introduces the Snort-Based Wi-Fi Module used for detecting malicious wireless network traffic aimed at PIDs. Section 4.3 describes the implementation of the Bluetooth Attack Detection and Signature System (BADSS) Module used for detecting Bluetooth attacks. Section 4.4 describes the Correlation Intrusion Detection Engine (CIDE) server which implements attack correlation, notification, and response.

4.1 B-SIPS Client

Because PIDs operate in mobile environments with limited battery and processing resources, conventional means of intrusion detection are not practical. In an attempt to remedy this situation, B-SIPS clients make use of a different approach to detect wireless attacks. The B-SIPS client [1], as shown in Figure 18, is a host-based application that continually monitors PID IC for anomalous behavior in the attempt to detect wireless attacks aimed at draining excessive battery power. Since the B-SIPS client is not as demanding of PID resources and shown effective by Buennemeyer [1], it is used as the triggering mechanism for MVP-IDS, as described in section 4.1.1. Section 4.1.2 discusses B-SIPS client status reporting. Section 4.1.3 details the cryptographic routines that have been added to the B-SIPS client and the CIDE server to secure message transactions. Section 4.1.4 investigates how the CIDE server now has the ability to warn clients of possible intrusions. Section 4.1.5 describes the layout of the B-SIPS client status report database and how it is used.

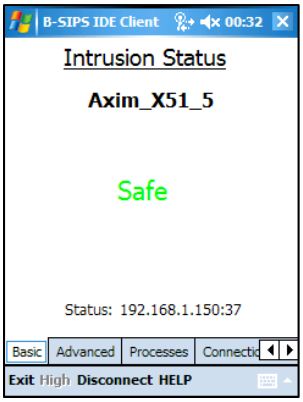


Figure 18. B-SIPS client.

4.1.1 Detecting and Correlating Attacks

As previously mentioned, the methodology of this research is to identify IC anomalies from B-SIPS clients and to also validate those reports with real-time Wi-Fi and Bluetooth attack traffic. While IC anomaly discovery was sufficiently implemented in Buennemeyer's implementation of B-SIPS, the correlation of actual attack traffic needed great improvement. In order to correlate IC anomalies with Bluetooth and Wi-Fi attack traffic, a method of obtaining and logging wirelessly transmitted packet streams had to be implemented. Section 4.1.1.1 discusses the theoretical approach to monitoring wireless packet streams and section 4.1.1.2 describes the implementation actually used.

4.1.1.1 Theoretical Approach

In conventional IDSs, it is common for packet streams to be continually monitored and analyzed to successfully perform attack detection. However, in the mobile environment, in which PIDs operate, resources for this continual task are not available when extended battery lifetimes are required. Because of stringent hardware constraints on PIDs, including: limited memory, restricted battery power, and lower CPU processing capabilities, conventional IDSs are not practical due to their high consumption rate of PID resources. With PID constraints taken into account, it was theorized to only have wirelessly transmitted packet streams analyzed once an IC threshold was breached, essentially using anomalies in IC as a triggering mechanism. Raw packet transmissions from Bluetooth and Wi-Fi would be logged to the PID while waiting on the IC trigger, with logs replacing themselves every *X* number of seconds or *Y* number of bytes. The logs, however, would only be examined by the CIDE server if an IC threshold was breached.

If raw packet transmissions were sent in their entirety to the CIDE server, network traffic would essentially double. With this possibly being a large overhead, tests were executed to determine if entire packets were needed for attack detection, or if only certain parts of each packet were needed. The results in section 5.2.1.1 and previous work from O'Connor [49] show that only Wi-Fi packet headers and high-level signaling Bluetooth packets are needed for attack detection. Therefore, this approach would only keep enough information in logs for attack detection by the CIDE server. This reduction of packet information saves Wi-Fi bandwidth, while not compromising attack detection and correlation.

Once an IC anomaly is detected, the B-SIPS client would then send its Bluetooth and Wi-Fi packet logs, along with battery data, to the CIDE server for further attack detection and correlation. The hypothesis of this theoretical approach is that IC anomalies could be validated through the correlation of malicious packet streams occurring in the same time interval and directed at the same PID. While this approach is feasible using a full-featured OS, it is not supported in the Windows Mobile environment and .NET Compact Framework [50].

4.1.1.2 Implemented Approach

Windows Mobile's lack of raw socket access to wireless packet steams forces the theorized approach to be simulated in the attempt to seek validation and predict feasibility. To simulate the theoretical idea, packet streams still had to be monitored, but by a full-featured operating system. The Snort-based Wi-Fi module and the BADSS module were developed because of this constraint. Both modules continually monitor packet streams in real-time and log attack discoveries into databases that can be queried by the CIDE server. Therefore, when the CIDE server receives status reports from B-SIPS clients that contain IC anomalies, it can then correlate the device status report with attacks logged by the Snort and BADSS attack databases. While this is simply a simulated approach used as a workaround, the theoretical approach seems highly feasible once a raw socket support is implemented in the mobile environment.

4.1.2 B-SIPS Client Status Reporting

B-SIPS clients continually monitor device state and operating characteristics at one second intervals. This allows for notable changes in device operating conditions, such as IC, to be detected and logged for further forensic analysis. Once a device has logged its state information for ten consecutive seconds, the optimum rate for battery preservation [1], it then sends its data to the CIDE server.

4.1.3 Securing Status Reporting

In Buennemeyer's implementation of status reporting from B-SIPS clients, all message transactions were transmitted in plaintext. No forms of cryptography were used to ensure that messages received by the CIDE server were confidential and received exactly as sent. Many attacks could possibly be used against B-SIPS client status reporting to modify or eavesdrop on message transactions. If this were to happen, it would compromise the MVP-IDS system as a whole. To prevent this scenario, messages have a set procedure in which they are formed and transmitted in MVP-IDS. B-SIPS client status reports are now assembled into messages of following format:

Message Format: **Field="value"**

ID="IP Address"**Nonce**="Nonce Value"**Data**="Encrypted Message"**Auth**="Authentication Check"

Message Field Explanation:

- **ID:** This field is used to identify the sender of the message.
- **Nonce:** Nonce stands for number used only once. It is a unique, ever-increasing, but not sequential, value used in the authentication check to thwart replay attacks. The most recent nonce for a sender ID is stored and used to evaluate the freshness of a message. Messages arriving that have a nonce value less than or equal to a previous nonce are invalid and disregarded.
- **Data:** This is the actual PID battery data of a particular device. It is encrypted using the RC2 .NET Compact Framework implementation to thwart man-in-the-middle attacks and prevent eavesdropping. The RC2 symmetric encryption algorithm was chosen for the MVP-IDS system because of its fast encryption/decryption times and energy-efficiency [51] [52].
- **Auth:** This field is used to check the integrity of the B-SIPS client status report. It assures that the original message is from the specified sender, has not been replayed, and that no additions, deletions or modification have been made. The MD5 secure hash algorithm implementation from the .NET Compact Framework is used in generating the Auth field authentication check. The Auth field value is also encrypted using the RC2 encryption routine for added security.

Message Value Explanation:

- **IP Address:** This value is the IP address of the sender in dot-decimal notation.
- **Nonce Value:** The nonce value is a number generated based on the current time of the sender system as an ever increasing, 64-bit integer.
- **Encrypted Message:** This value is the base64 encoding of the RC2 encryption of the B-SIPS client status report. The following is a Pseudo-code implementation:

  base64encode(RC2_Encrypt(B-SIPS battery data)))

- **Authentication Check:** This value is the base64 encoding of the RC2 encrypted MD5 hash of the concatenation of the B-SIPS client report, the Nonce Value, and the IP Address. The following is a Pseudo-code implementation:

  base64encode(RC2_Encrypt(MD5_Hash(B-SIPS battery data + Nonce Value + IP Address)))

With the implemented message formatting and added cryptographic routines, B-SIPS client reports are much more secure and less susceptible to attack. The CIDE server can now be assured that data received from a B-SIPS client is exactly what was originally sent, and that all message transactions are confidential between the two systems.

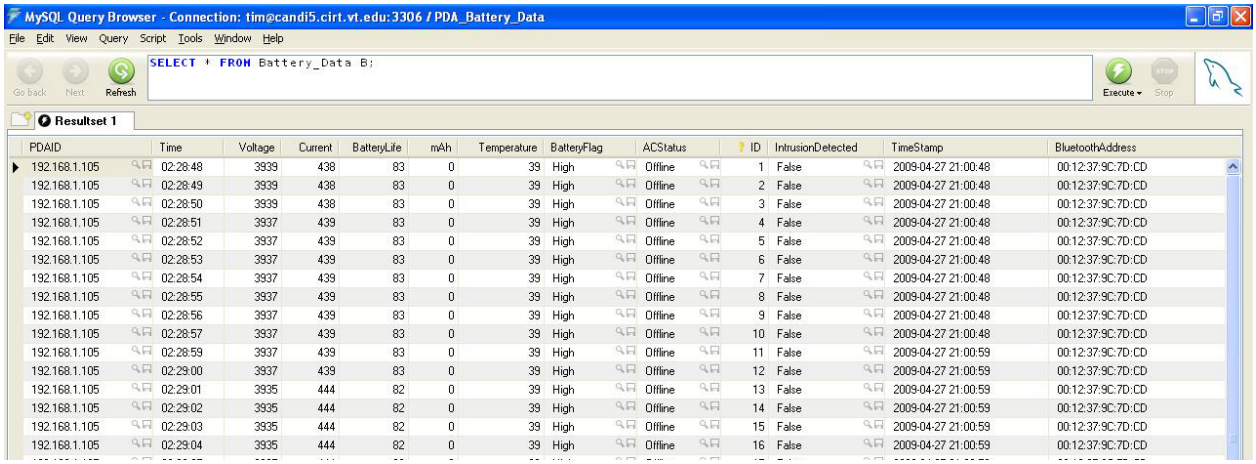
4.1.4 Administrative Response Receiving

MVP-IDS also implements an administrative response. These are reactionary messages sent to B-SIPS clients deemed under attack from malicious wireless communications. This is possible because of the bi-directional communication system included within MVP-IDS. Unlike the original B-SIPS project that was uni-directional, B-SIPS clients and the CIDE server are now able to effectively communicate with one another.

Administrative responses provide feedback to users about the attack being launched against them, supplies counter measures that the CIDE server is taking to mitigate the attack, and only disables the wireless interface being attacked. The administrative response mechanism is detailed further in section 4.4.4.3, as it is an operational feature of the CIDE server.

4.1.5 B-SIPS Client Database Logging

All B-SIPS client status reports are stored in a MySQL back-end database. This is done to allow for forensic analysis by SA's and attack correlation by the CIDE server. Figure 19 shows the format of the B-SIPS client status report database. The only change that has been made to the B-SIPS client status report database from Buennemeyer's implementation is the addition of a Bluetooth device address field.



PDAID	Time	Voltage	Current	BatteryLife	mAh	Temperature	BatteryFlag	ACStatus	ID	IntrusionDetected	TimeStamp	BluetoothAddress
192.168.1.105	02:28:48	3939	438	83	0	39	High	Offline	1	False	2009-04-27 21:00:48	00:12:37:9C:7D:CD
192.168.1.105	02:28:49	3939	438	83	0	39	High	Offline	2	False	2009-04-27 21:00:48	00:12:37:9C:7D:CD
192.168.1.105	02:28:50	3939	438	83	0	39	High	Offline	3	False	2009-04-27 21:00:48	00:12:37:9C:7D:CD
192.168.1.105	02:28:51	3937	439	83	0	39	High	Offline	4	False	2009-04-27 21:00:48	00:12:37:9C:7D:CD
192.168.1.105	02:28:52	3937	439	83	0	39	High	Offline	5	False	2009-04-27 21:00:48	00:12:37:9C:7D:CD
192.168.1.105	02:28:53	3937	439	83	0	39	High	Offline	6	False	2009-04-27 21:00:48	00:12:37:9C:7D:CD
192.168.1.105	02:28:54	3937	439	83	0	39	High	Offline	7	False	2009-04-27 21:00:48	00:12:37:9C:7D:CD
192.168.1.105	02:28:55	3937	439	83	0	39	High	Offline	8	False	2009-04-27 21:00:48	00:12:37:9C:7D:CD
192.168.1.105	02:28:56	3937	439	83	0	39	High	Offline	9	False	2009-04-27 21:00:48	00:12:37:9C:7D:CD
192.168.1.105	02:28:57	3937	439	83	0	39	High	Offline	10	False	2009-04-27 21:00:48	00:12:37:9C:7D:CD
192.168.1.105	02:28:59	3937	439	83	0	39	High	Offline	11	False	2009-04-27 21:00:59	00:12:37:9C:7D:CD
192.168.1.105	02:29:00	3937	439	83	0	39	High	Offline	12	False	2009-04-27 21:00:59	00:12:37:9C:7D:CD
192.168.1.105	02:29:01	3935	444	82	0	39	High	Offline	13	False	2009-04-27 21:00:59	00:12:37:9C:7D:CD
192.168.1.105	02:29:02	3935	444	82	0	39	High	Offline	14	False	2009-04-27 21:00:59	00:12:37:9C:7D:CD
192.168.1.105	02:29:03	3935	444	82	0	39	High	Offline	15	False	2009-04-27 21:00:59	00:12:37:9C:7D:CD
192.168.1.105	02:29:04	3935	444	82	0	39	High	Offline	16	False	2009-04-27 21:00:59	00:12:37:9C:7D:CD

Figure 19. B-SIPS client status report database.

4.2 Snort-Based Wi-Fi Module

As mentioned in section 4.1.1: detecting and correlating attacks, the original approach to detecting Wi-Fi attacks on PIDs was to continually log packet streams on each B-SIPS client and then have the CIDE server analyze the log for attacks. However, it was determined that this method was infeasible due to the lack of a raw sockets implementation by Microsoft. As a result of this obstacle, this research implemented a real-time Wi-Fi traffic monitoring module to allow for IC anomalies from B-SIPS clients to be correlated with attacking Wi-Fi packet streams. Section 4.2.1 discusses network-based attacks transmitted via Wi-Fi and the implications on target PIDs. Section 4.2.2 introduces sockets and how the use of raw sockets were planned to be an integral part of this research. Section 4.2.3 investigates signature-based IDSs and their components. Section 4.2.4 details the design and methodology of the Snort-based Wi-Fi Module. Section 4.2.5 describes the MySQL database that attack records are logged to once they are detected.

4.2.1 Network-Based Attacks

While the range of network-based attacks against standard PCs varies greatly from DoS to SQL injections, the range of attacks targeting PIDs is far more specific and specialized. This research effort primarily focused on the need to protect PID and their battery lifetimes from DoS and reconnaissance attacks. These classifications are made as follows:

- **Denial of Service (DoS):** Network-based attacks of this nature are used to flood network traffic to target PIDs. The goal of DoS flooding attacks is to specifically craft packets and repeatedly send them to target PIDs at high packet per second rates with the intent of battery exhaustion.
- **Reconnaissance:** These attacks are aimed at recovering device information that could possibly uncover device weaknesses and vulnerabilities. Attacks of this type are usually run once, as this is all that is needed for an attacker to gain the information needed from their target.

From the above classifications, a list of 14 different attacks was assembled to gain insight on PID attack response and to test MVP-IDS. Table 5 shows the attacks and their associated classifications.

Table 5. Network-based attacks deployed against PIDs.

#	Attack Name	Category
1	Ping Flood	DoS
2	ACK Flood	DoS
3	FIN Flood	DoS
4	PUSH Flood	DoS
5	RST Flood	DoS
6	SYN Flood	DoS
7	URG Flood	DoS
8	XMAS Flood	DoS
9	YMAS Flood	DoS
10	Nessus Default Scan	Reconnaissance
11	Nmap Intense Scan	Reconnaissance
12	Nmap OS Scan	Reconnaissance
13	Nmap Quick Scan	Reconnaissance
14	Unicorn Scan	Reconnaissance

4.2.2 Sockets

Developed into an Application Programming Interface (API) library, sockets were first introduced by the University of California, Berkeley in 1980. Since then, they have become the standard implementation for all network programming. A socket is simply one endpoint of a network communications link between two applications [53]. It specifies both a computer system and the communicating application by using a unique IP address and a unique port number. This allows for a multi-system/multi-application environment in which networked computer systems can easily communicate.

Based upon the Berkeley sockets API, Internet domain raw sockets allow raw data, or actual byte transmissions from Ethernet frames, to be forwarded and used by an application [13]. If an application has access to transmitted frames, it is also able to extract network, transport, and other high-level protocol header information from the frame's data field, as shown in Figure 20 [54] [55]. Raw sockets can be used for good intentions, such as designing IDSs and penetration testing suites that security-harden computer systems. Conversely, raw sockets also allow users to craft and send specialized packets with manipulated headers for IP spoofing and other malevolent intents. Having access to such low-level data is a very powerful privilege in the computing environment and was a hopeful addition originally proposed for the B-SIPS client. However, this idea had to be deserted once Microsoft confirmed that raw sockets were not a part of the .NET Compact Framework, even after misleading documentation stated that they were. If raw sockets were a possibility, an implementation of our theoretical approach to packet capturing and CIDE server analysis seems highly feasible.

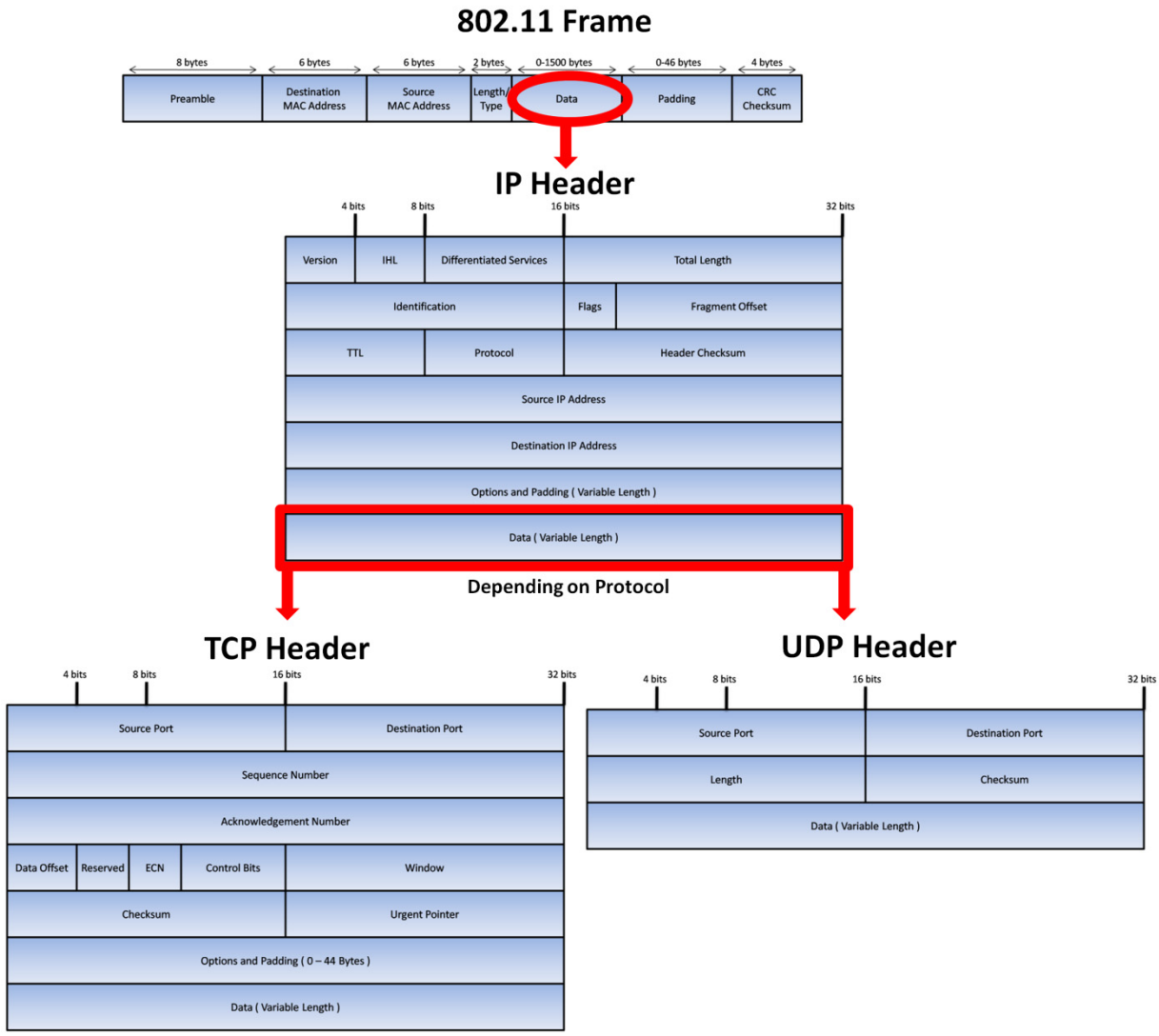


Figure 20. Network and transport layer header extraction from the Ethernet frame data field.

4.2.3 Detecting Attacks

In order for Snort or any other IDS to detect Wi-Fi attacks, it must be able to process all network traffic that passes by its sensors. To do so, an IDS uses a raw sockets implementation to analyze all data sent and received on a network link. As the traffic passes the sensor, the IDS tries to match the data being transmitted with that of known attack signatures. According to Symantec, "an attack signature is a unique arrangement of information that can be used to identify an attacker's attempt to exploit a known operating system or application vulnerability[56]". Signature-based IDSs include attack signature databases which contain all known attack signatures that the IDS is capable of detecting. As attack signatures are defined and developed, they can then be added to the signature database to allow for further attack detection.

4.2.4 Module Design and Overview

Since Snort is highly used in industry and recognized as a quality IDS, it was chosen to analyze Wi-Fi network streams between B-SIPS clients and other computing devices. The main objective of the Snort-based module was to ensure that all Wi-Fi traffic had the ability to be monitored and analyzed. To do this, the module was designed to satisfy the following four goals:

1. All B-SIPS clients need an active network connection to effectively transmit B-SIPS device status reports to the CIDE server.
2. A subnet must be designed such that bi-directional communication between B-SIPS client and the CIDE server is possible.
3. A Snort sensor must be able to analyze all Wi-Fi traffic to and from B-SIPS clients.
4. The design of the subnet and Snort sensor must not hinder B-SIPS client status reporting in anyway, in effect, being transparent to all B-SIPS client PIDs and the CIDE server.

All four goals were successfully implemented in the Snort-Based Wi-Fi Module. The design begins with an active Internet connection from an ISP switch to the MVP-IDS subnet. The Snort sensor was then placed between the MVP-IDS subnet switch and a WAP. To allow the Snort sensor to be transparent and still effectively detect Wi-Fi attacks, a network tap was used to passively sniff all transmitted Wi-Fi traffic. The network tap simply splits traffic signals to the Snort sensor, via a bond between two network interface cards, and to its destination without hindering any subnet communications. Figure 21 details the Snort-Based Wi-Fi Module and the interactions with each of its components.

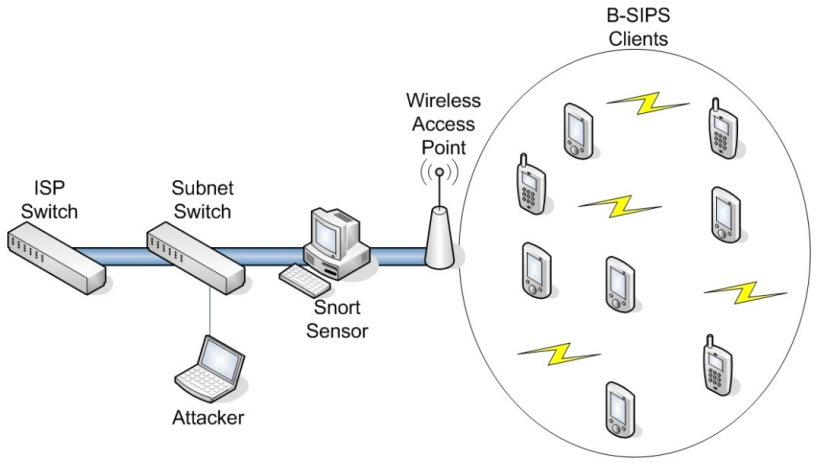


Figure 21. Snort-based Wi-Fi module.

4.2.3 Snort Module Database Logging

Much like the B-SIPS client status report database, all Snort attack records are stored in a MySQL database. This is done to give the CIDE server the ability to correlate B-SIPS client status reports containing attacks with any attacks that were logged from the Snort sensor. Figure 22 shows the format of the Snort attack database.

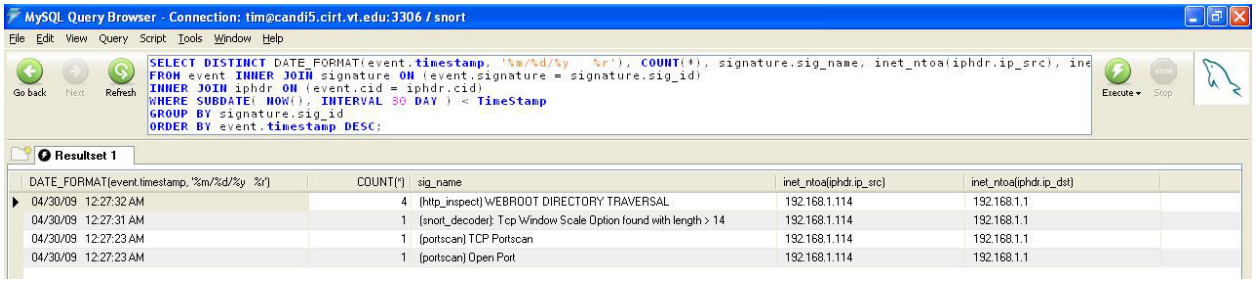


Figure 22. Snort attack database.

4.3 Bluetooth Attack Detection and Signature System (BADSS) Module

One of the shortcomings of Buennemeyer's implementation of B-SIPS is that there was no practical way to correlate IC anomalies to Bluetooth attack traffic. BADSS attempts to address this problem by developing a separate module to monitor and detect malicious Bluetooth communications. As mentioned in previous sections, the lack of raw sockets was a severe hindrance to the development of this research, as it was the theoretical approach to all real-time packet monitoring for Wi-Fi and Bluetooth traffic alike. Since there is also not an implementation for Bluetooth raw sockets in the .NET Compact Framework, an alternative approach had to be used for the BADSS module as well. Supplementing O'Connor's previous work [49], BADSS implements the second known Bluetooth IDS for signature matching attacks from recorded Bluetooth packet streams. Section 4.3.1 discusses a Bluetooth attack classification framework to better understand Bluetooth attacks. Section 4.3.3 explores the implementation and intricacies of the BADSS module. Section 4.3.4 details the logging of Bluetooth attack records to a MySQL database.

4.3.1 Bluetooth Attack Classification Framework

Since Bluetooth is a relatively new technology and protocol specification, all vulnerabilities and implementation flaws have not been discovered or mitigated at this point in time. Because of this, attackers have been able to exploit devices through the Bluetooth medium. This research has assembled a list of 21 common Bluetooth attacks, as shown in Table 6, and implemented a framework to classify each attack for better understanding. Each attack is classified using the following three components: Name, Focus, and Exploit.

- **Name:** This is the common, or general, name given to an attack.
- **Focus:** This classification is used to describe the intent of an attack. Reasons for attacking a device vary primarily depending on what the attacker gains by successfully exploiting a target device. This research proposes to categorize Bluetooth attack focuses into 4 distinct categories:
    1. **Device Discovery:** Attacks of this genre focused on discovering devices in range of attack and obtaining their Bluetooth device addresses. This type of attack is mainly used as a passive reconnaissance tool to gain valuable information about a target device before actually attacking it. In non-discoverable mode, a device never responds to an inquiry scan or request. Therefore, if an attacker wants to attack a device in non-discoverable mode, it must first obtain the Bluetooth

device address of the target somehow. Examples of tools that implement Bluetooth device discovery are RedFang, Btscanner, Tbear.

2. **Service Discovery:** Attacks from this category are aimed at obtaining the types of Bluetooth services that a target device is capable of performing. Similar to device discovery attacks, service discovery attacks gather information about a target. Examples of tools that implement Bluetooth service discovery attacks are BluePrint, PSM Scan, and RFCOMM Scan.

3. **Information Theft:** Many people keep confidential information on PIDs that could cause hardships to the victim if stolen. This type of attack focuses on infiltrating the device to steal a user's confidential information. Examples of tools that implement Bluetooth information theft attacks are BlueBug, BlueSnarf, Btcrack, CarWhisperer, and Helomoto.

4. **Denial of Service (DoS):** DoS attacks are used to render some service or resource of a device unavailable. This could mean temporarily denying a service, or causing a device to malfunction requiring a reset of the entire system. Attacks of this nature usually target a flaw in a software implementation of the protocol or specification. Examples of tools that implement Bluetooth DoS attacks are BlueSmack, Nasty vCard, L2CAP Header Overflow, HCIDumpCrash, Nokia N70 DoS, Bluetooth Stack Smasher, Ping of Death, Tanya, BlueSpam, and a new attack developed in this research, Blueper.

- **Exploit:** Attacks are successful because attackers have found a weakness in the target device and use that weakness as an exploitable vulnerability. This research proposes to categorize Bluetooth exploits into 6 distinct categories:

    1. **Bluetooth Specification:** This classification is given to an attack most often focused on device discovery and targets device information reconnaissance. There is no exact exploitation flaw, per say, success of this type of attack is based solely on the protocols and operations set forth for communication in the Bluetooth specification. Attacks that exploit the Bluetooth specification are usually aimed at gathering device target addresses, so that the attacker may launch more sophisticated and devastating attacks. Examples of Bluetooth attacks that exploit the Bluetooth specification are RedFang, BTScanner, and Tbear.

    2. **Device Discovery:** Typically, attacks in this category are successful primarily due a target device being discovered. Bluetooth users can mitigate this exploit by turning off Bluetooth services when not in use or keeping the Bluetooth device in non-discoverable mode during normal operations. Although Bluetooth devices can be discovered when in non-discoverable mode, it is much more difficult and requires much more effort for an attacker. Attacks that exploit device

discovery are usually aimed at either further reconnaissance of information about a target device through service discovery or launching DoS attacks. Examples of Bluetooth attacks that exploit device discovery are BluePrint, PSM Scan, RFCOMM Scan, BlueSpam, and Blueper.

3. **Authentication:** This exploit is mostly directed at devices deployed with implementations of the Bluetooth specification prior to version 2.1. The Bluetooth specification, prior to version 2.1, has known weaknesses in the pairing and authentication process which was intended to allow devices to securely communicate. Attacks that exploit weak versions of the Bluetooth authentication process are usually focused on information theft, giving attackers access to what victim devices thought were secure communication channels. Examples of Bluetooth attacks that exploit the authentication process are BlueBug, BlueSnarf, BTCrack, CarWhisperer, and Helomoto.

4. **Buffer Overflow:** A buffer overflow is a software implementation flaw which arises when a process attempts to store data outside of the intended memory range that a developer allocated for it. This exploit has nothing to do with the Bluetooth specification or weaknesses in it. It is directly related to flaws in the software implementation, mainly because of programmers neglecting to validate user input or packet payloads that are received. Examples of Bluetooth attacks that exploit target devices through buffer overflows are BlueSmack and Nasty vCard.

5. **Malformed Packets:** Much like buffer overflows, malformed packets cause devices to malfunction because of software implementations of the Bluetooth specification that do not properly validate user input or packets that are received before packet data is processed. Attacks that implement this exploit are usually more sophisticated and require an attacker to build Bluetooth packets with invalid parameters. This is usually accomplished through the use of out-of-range signaling commands or invalid signal lengths. Examples of Bluetooth attacks that exploit targets by sending malformed packets are L2CAP Header Overflow, HCIDumpCrash, Nokia N70 DoS, and Bluetooth Stack Smasher.

6. **Resource Consumption:** This exploit is only applicable when an attacker is launching a DoS attack. The Bluetooth specification sets standard ways that devices respond to certain types of packets and protocol services. From this, attackers can then consume device resources or Bluetooth bandwidth, by continually flooding target devices with requests for resources. Attacks of this nature are primarily used if an attacker wants to hinder Bluetooth communication with other devices or to exhaust PID batteries. Examples of Bluetooth attacks that exploit targets through resource consumption are Ping of Death and Tanya.

Table 6. Bluetooth attack classifications.

#	Attack	Focus	Exploit
1	RedFang	Device Discovery	Bluetooth Specification
2	Btscanner	Device Discovery	Bluetooth Specification
3	Tbear	Device Discovery	Bluetooth Specification
4	BluePrint	Service Discovery	Device Discovery
5	PSM Scan	Service Discovery	Device Discovery
6	RFCOMM Scan	Service Discovery	Device Discovery
7	BlueBug	Information Theft	Authentication
8	BlueSnarf	Information Theft	Authentication
9	Btcrack	Information Theft	Authentication
10	CarWhisperer	Information Theft	Authentication
11	Helomoto	Information Theft	Authentication
12	BlueSmack	Denial of Service	Buffer Overflow
13	Nasty vCard	Denial of Service	Buffer Overflow
14	L2CAP Header Overflow	Denial of Service	Malformed Packets
15	HCIDumpCrash	Denial of Service	Malformed Packets
16	Nokia N70 DoS	Denial of Service	Malformed Packets
17	Bluetooth Stack Smasher	Denial of Service	Malformed Packets
18	Ping of Death	Denial of Service	Resource Consumption
19	Tanya	Denial of Service	Resource Consumption
20	BlueSpam	Denial of Service	Device Discovery
21	Blueper	Denial of Service	Device Discovery

4.3.3 BADSS Module Design and Overview

The BADSS Module was built to recognize Bluetooth attacks and was designed consisting of two main components:

1. The Merlin II Bluetooth protocol analyzer was used for Bluetooth packet capturing and exporting of those captures to text files.

2. The BADSS Intrusion Detection Engine (IDE) processes each textual packet capture file, attempting to match the file's Bluetooth traffic patterns with attack signatures contained in its signature database.

Section 4.3.3.1 discusses the Merlin II Bluetooth protocol analyzer and its use in Bluetooth packet capturing. Section 4.3.3.2 presents the idea of Bluetooth attack signatures and also the new attack signatures developed

directly from this research effort. Section 4.3.3.3 explores the BADSS IDE and its use in Bluetooth intrusion detection. The BADSS module and all of its interacting components can be seen in Figure 23.

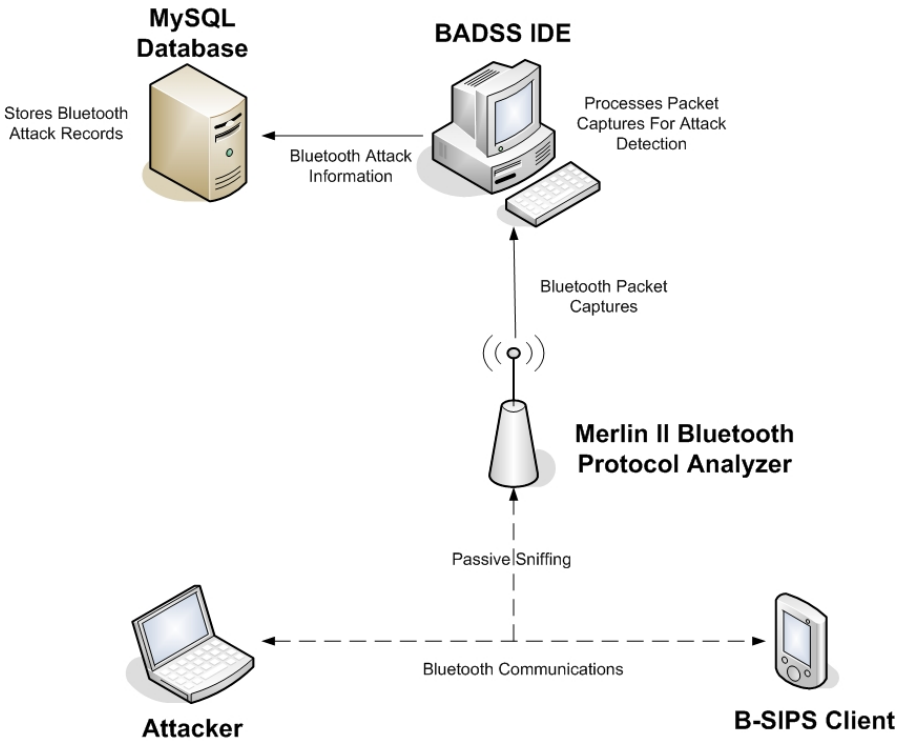


Figure 23. BADSS module.

4.3.3.1 Bluetooth Packet Capturing

To understand Bluetooth attacks, one must be able to see them decomposed all the way to the packet level. It is difficult to defend against an attack without being able to observe the communication between two devices during the attack. To aid in this understanding, this research utilized the Merlin II Bluetooth Protocol Analyzer, which allowed for viewing of packet-level Bluetooth communications. Not only does the Merlin II capture Bluetooth transmissions on the packet level, but it also has a preprocessor engine that assembles multiple low-level packets into a single high-level protocol packet, such as an LMP or an L2CAP packet. An example of a Merlin II packet capture and high-level protocol packet reassembly is shown in Figure 24.

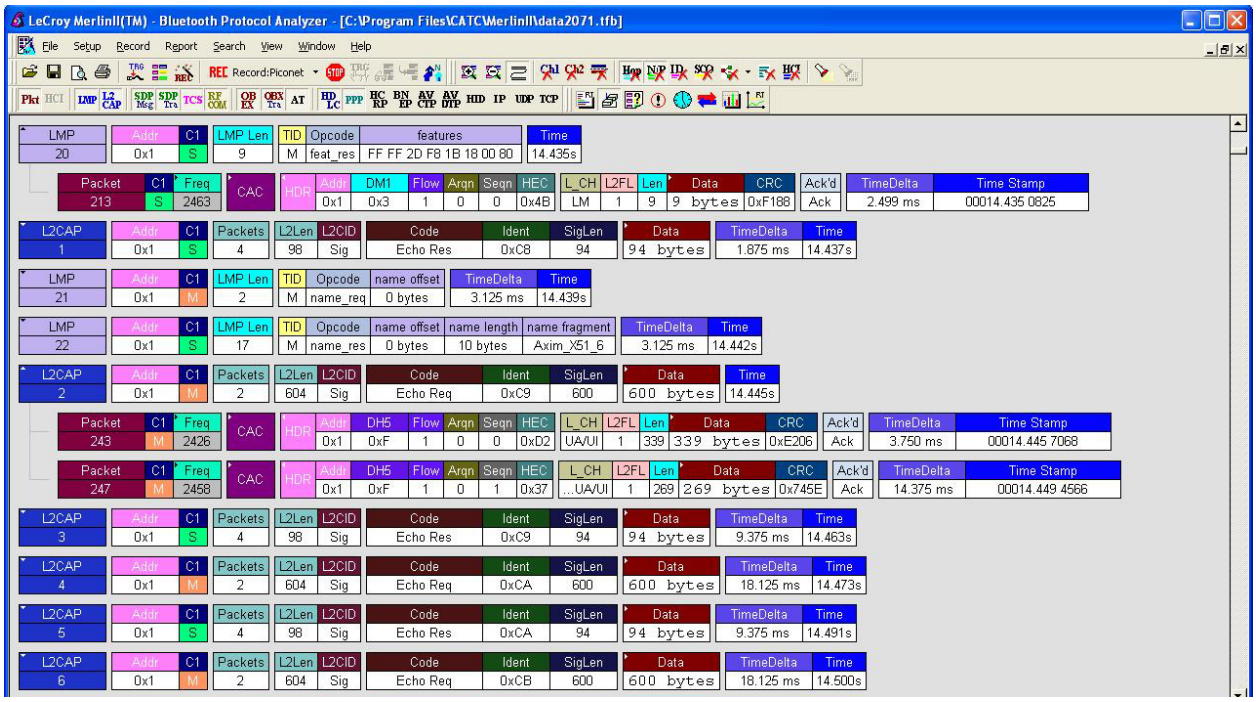


Figure 24. Merlin II screen shot of BlueSmack packet capture.

4.3.2.2 Bluetooth Attack Signatures

In order to develop a signature-based IDS for recognizing Bluetooth attacks, signatures for common attacks must be created. Previously, [49] developed packet-based attack signatures for all but two attacks listed in Table 6, those being *RedFang* and *Blueper*. These two new attacks, as well as their associated attack signatures, are described below.

1. **RedFang:** This device discovery tool is used to find Bluetooth-enabled devices that are operating in non-discoverable mode in close proximity of an attacker. It is a brute force tool that sequentially scans Bluetooth device addresses repeatedly; sending name requests to each address in a user-specified Bluetooth device address range. If a device responds to a name request, the attacker then knows that there is a device nearby with the associated Bluetooth device address. When a target device responds to a name request, RedFang then sends version and feature requests to the target device to provide even more valuable information to the attacker. Once an attacker has the target's Bluetooth device address, as well as its associated device information, a more sophisticated and harmful attack can be launched at the target. The signature for a RedFang attack is described below and shown in Figure 25:

Step:	Device	Action
1:	Attacker (Master)	Sends name request to target.
2:	Target (Slave)	Replies with name response.
3:	Attacker	Sends detach command terminating the connection.
4:	Attacker	Sends version request.
5:	Target	Replies with version response.
6:	Attacker	Sends feature request.
7:	Target	Replies with feature response.

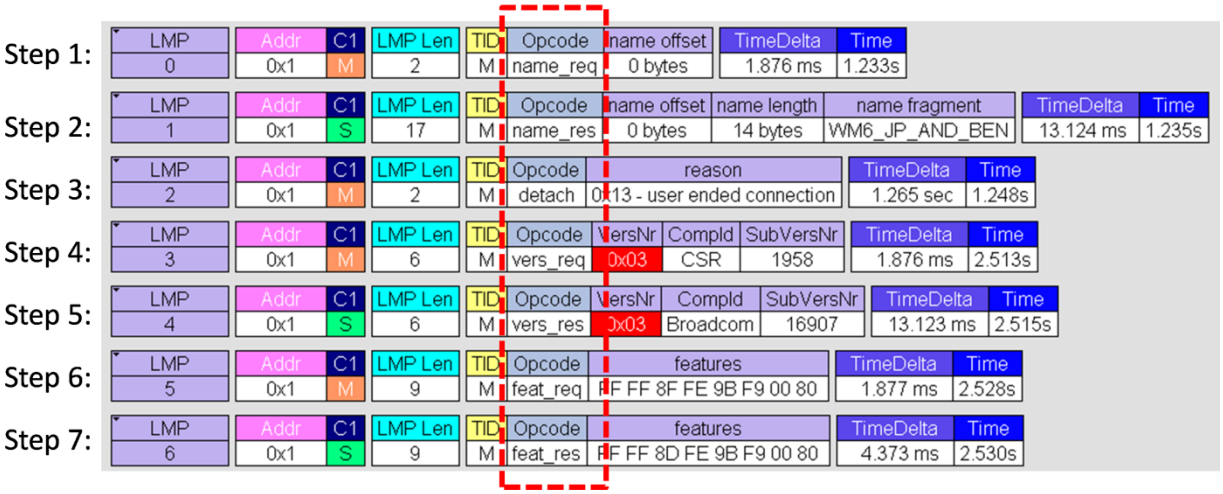


Figure 25. RedFang attack signature.

All seven communication steps between the attacker and target are accomplished in 1.6 seconds or less. This time was determined by repeated testing and analysis of RedFang packet captures. Many benign Bluetooth communication captures legitimately contain these packet transactions for valid communications. Therefore, in order to recognize a RedFang attack, timing is critical.

2. **Blueper:** A newly developed attack specifically for this research effort, Blueper aims to exhaust a target PID's battery and memory resources. To do so, it uses the *USSP-Push* tool to flood the target with incoming files. The file transfer prompts the user for interaction, but simultaneously in the background, stores the file in memory until the user confirms or denies the file. Even if the user denies a file, the attack is unaffected because another file transfer has already been started. The attacker continues to send files to the user until the target PID is out of range, turns off its

Bluetooth radio, or the PID battery has been completely exhausted. The signature for a Blueper attack is described below and shown in Figure 26:

Step:	Device	Action
1:	Attacker	Sends connection request to target.
2:	Target	Replies with connection response.
3:	Attacker	Sends configure request.
4:	Target	Replies with configure response.
5:	Attacker	Sends OBEX service search pattern.
6:	Attacker	Sends file through RFCOMM protocol.
7:	Attacker	Repeat Step 6.

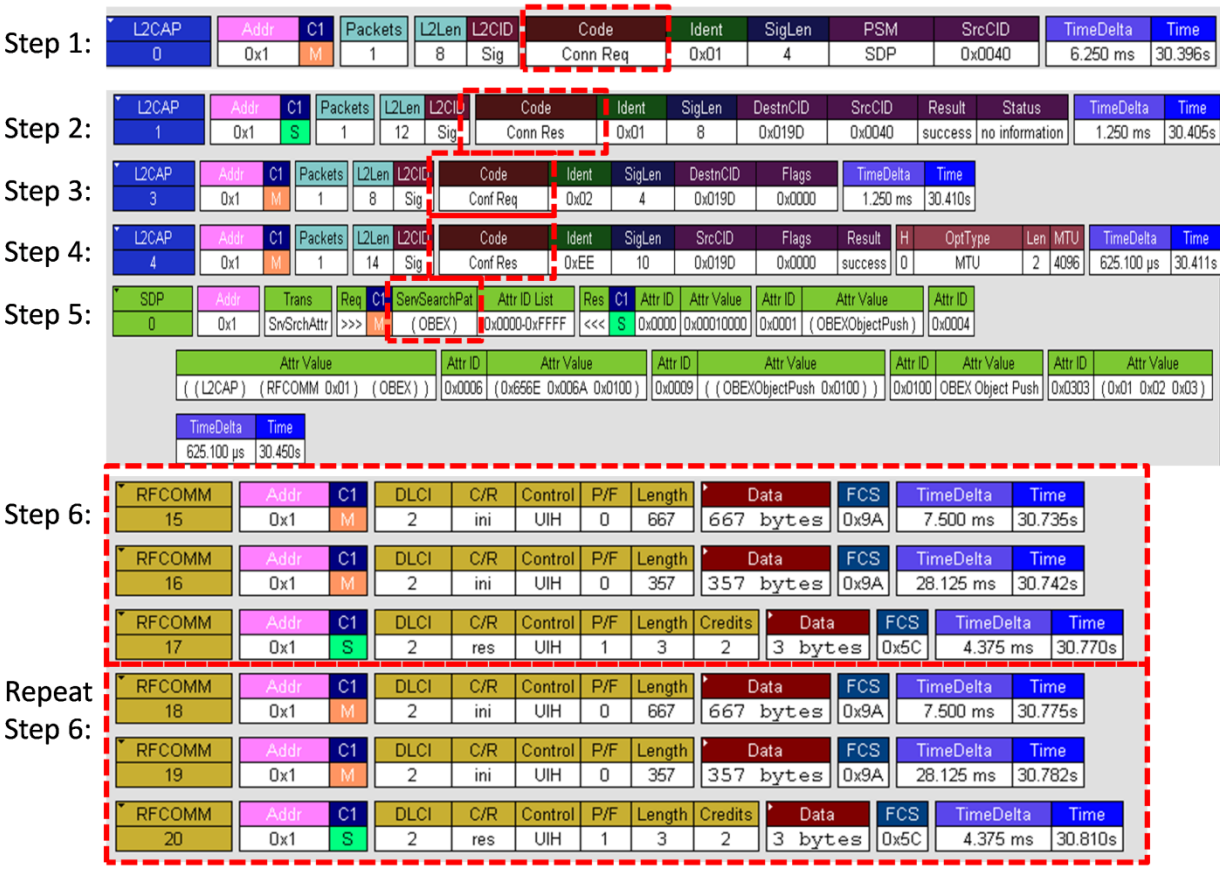


Figure 26. Blueper attack signature.

With steps 1 through 5 being simply connection setup, they are easy to recognize. However, steps 6 and 7 prove to be significantly more difficult to detect. Since files sent from the attacker can be any size, the signature for the Blueper attack is very generic. It operates on the basis of pattern matching, which attempts to look for repeated sets of packet transmissions between the attacker and target devices that are identical. Once the

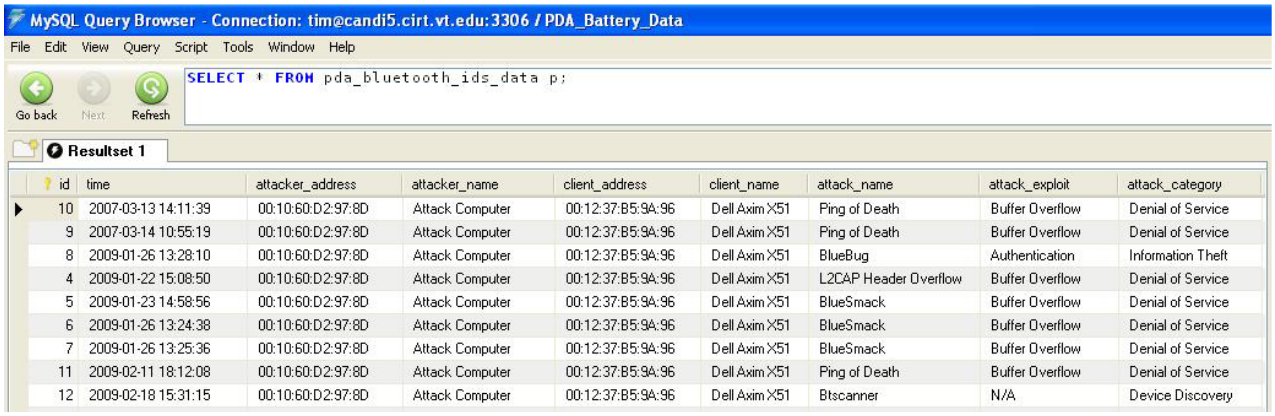
threshold for a set number of repeated file transfer patterns has been met, the attack signature has been successfully matched.

4.3.2.3 BADSS Intrusion Detection Engine (IDE)

The BADSS IDE is the actual application that processes Bluetooth packet streams and contains attack signatures for all attacks listed in Table 6. It accepts an exported text file containing a Bluetooth packet stream from the Merlin II Bluetooth protocol analyzer as input. Then, it parses each packet from the text file and stores it in a packet list. Next, the BADSS IDE attempts to find matches between traffic in the packet list and attack signatures stored in its attack signature database. If a match is discovered, the BADSS IDE inserts an attack record into the BADSS attack database.

4.3.3 BADSS Database Logging

All BADSS attack records are stored in a MySQL database. The BADSS attack database is implemented exactly in the same fashion as the Snort attack database; all attacks recognized by the BADSS IDE are logged into the database to be used by the CIDE server when performing real-time attack correlation. Figure 27 shows the format of the BADSS database.



id	time	attacker_address	attacker_name	client_address	client_name	attack_name	attack_exploit	attack_category
10	2007-03-13 14:11:39	00:10:60:D2:97:8D	Attack Computer	00:12:37:B5:9A:96	Dell Axim X51	Ping of Death	Buffer Overflow	Denial of Service
9	2007-03-14 10:55:19	00:10:60:D2:97:8D	Attack Computer	00:12:37:B5:9A:96	Dell Axim X51	Ping of Death	Buffer Overflow	Denial of Service
8	2009-01-26 13:28:10	00:10:60:D2:97:8D	Attack Computer	00:12:37:B5:9A:96	Dell Axim X51	BlueBug	Authentication	Information Theft
4	2009-01-22 15:08:50	00:10:60:D2:97:8D	Attack Computer	00:12:37:B5:9A:96	Dell Axim X51	L2CAP Header Overflow	Buffer Overflow	Denial of Service
5	2009-01-23 14:58:56	00:10:60:D2:97:8D	Attack Computer	00:12:37:B5:9A:96	Dell Axim X51	BlueSmack	Buffer Overflow	Denial of Service
6	2009-01-26 13:24:38	00:10:60:D2:97:8D	Attack Computer	00:12:37:B5:9A:96	Dell Axim X51	BlueSmack	Buffer Overflow	Denial of Service
7	2009-01-26 13:25:36	00:10:60:D2:97:8D	Attack Computer	00:12:37:B5:9A:96	Dell Axim X51	BlueSmack	Buffer Overflow	Denial of Service
11	2009-02-11 18:12:08	00:10:60:D2:97:8D	Attack Computer	00:12:37:B5:9A:96	Dell Axim X51	Ping of Death	Buffer Overflow	Denial of Service
12	2009-02-18 15:31:15	00:10:60:D2:97:8D	Attack Computer	00:12:37:B5:9A:96	Dell Axim X51	Btscanner	N/A	Device Discovery

Figure 27. BADSS attack database.

4.4 Correlation Intrusion Detection Engine (CIDE) Server

The CIDE server is a graphical user interface application used as the centerpiece of the MVP-IDS design. It acts as a centralized site for B-SIPS clients to send detailed reports describing the status of the PID and associated smart battery data. Upon B-SIPS clients reporting IC anomalies, the CIDE server attempts to correlate the anomalies with malicious Wi-Fi or Bluetooth communications to/from the PID. To allow SAs to quickly and efficiently monitor this process, the CIDE server is assembled into five separate views: Live Data, Database, Analysis, Data Correlation, and Device Profiles [1]. Each CIDE server view is described in sections 4.4.1 - 4.4.5, respectively.

4.4.1 Live Data View

The CIDE server Live Data View, shown in Figure 28, functions exactly the same way it did in the original B-SIPS implementation [1]. SAs use this view to monitor B-SIPS client status reports in a dynamically and constant updating fashion. As each device sends data to the CIDE server, it is logged in this view to allow SAs to quickly see how PID characteristics are changing in real-time. When the CIDE server receives a B-SIPS client status report indicating an IC anomaly, it can be easily seen in this view since all B-SIPS client status reports flagged with anomalous behavior are highlighted in red.

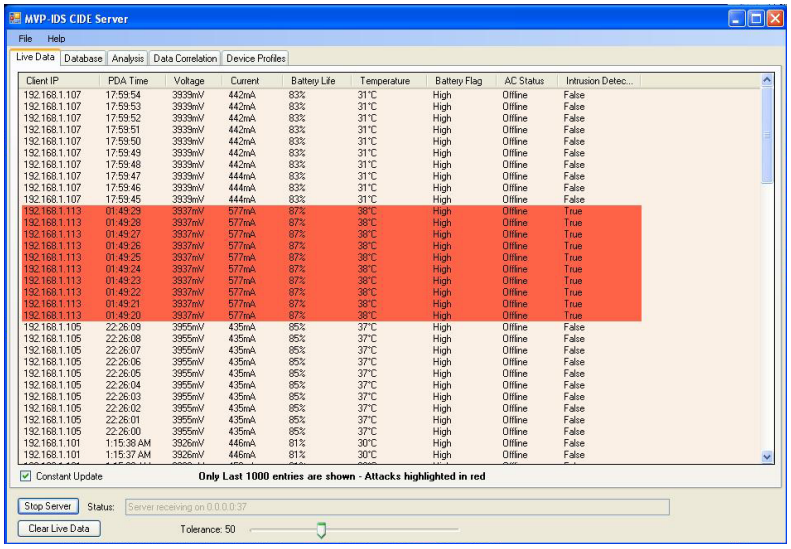


Figure 28. MVP-IDS CIDE server Live Data View.

4.4.2 Database View

The CIDE server Database View, shown in Figure 29, is used for displaying data stored in the B-SIPS client status report database, Snort attack database, BADSS attack database, and a blended attack database. This view has changed from Buennemeyer's implementation due to the increase in number of databases which store critical information. Buennemeyer's implementation of the CIDE server only included data from two databases: a B-SIPS client status report database and a Snort attack database. Changes were made to the Database view to correctly represent the increased number of databases and also give the SA more information when performing forensic analysis.

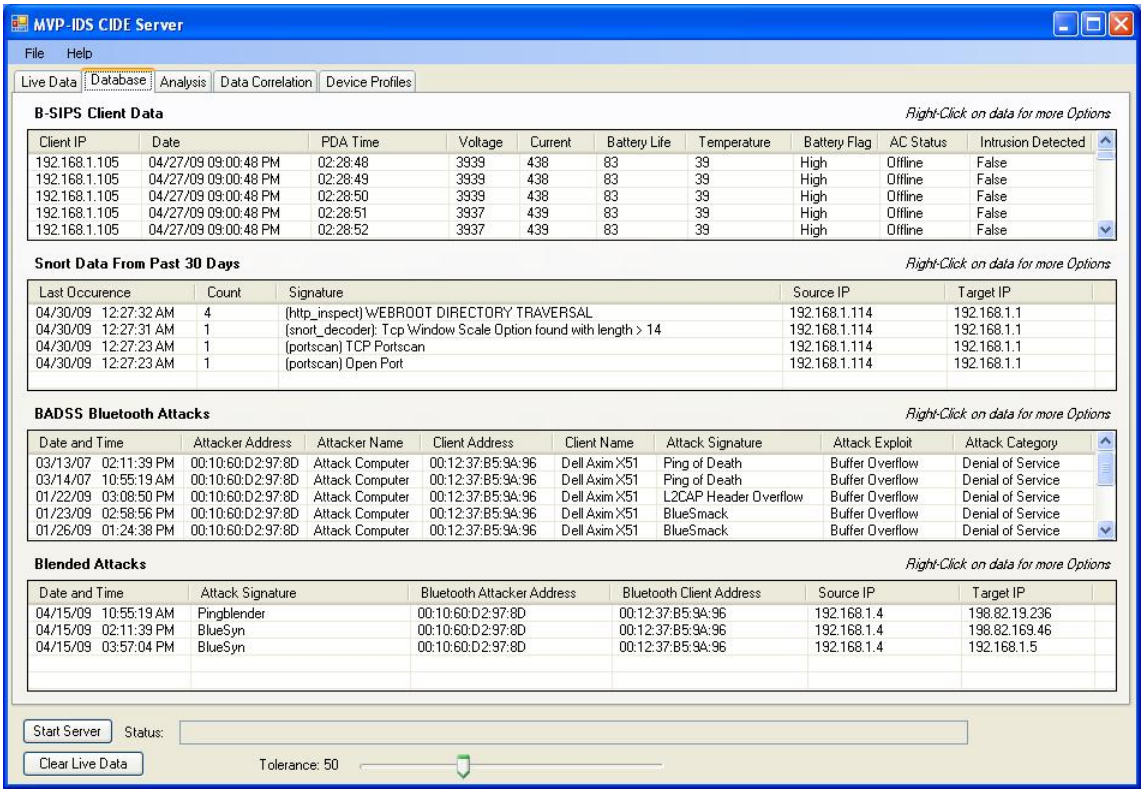


Figure 29. MVP-IDS CIDE server Database View.

4.4.3 Analysis View

The Analysis View, shown in Figure 30, is used to graphically display two two-dimensional graphs: battery current versus time and battery voltage versus time. These graphs are dynamically generated and updated as B-SIPS client status reports are received from B-SIPS client PIDs. Changes have not been made to this view of the CIDE server, as this was not in the scope of this research.

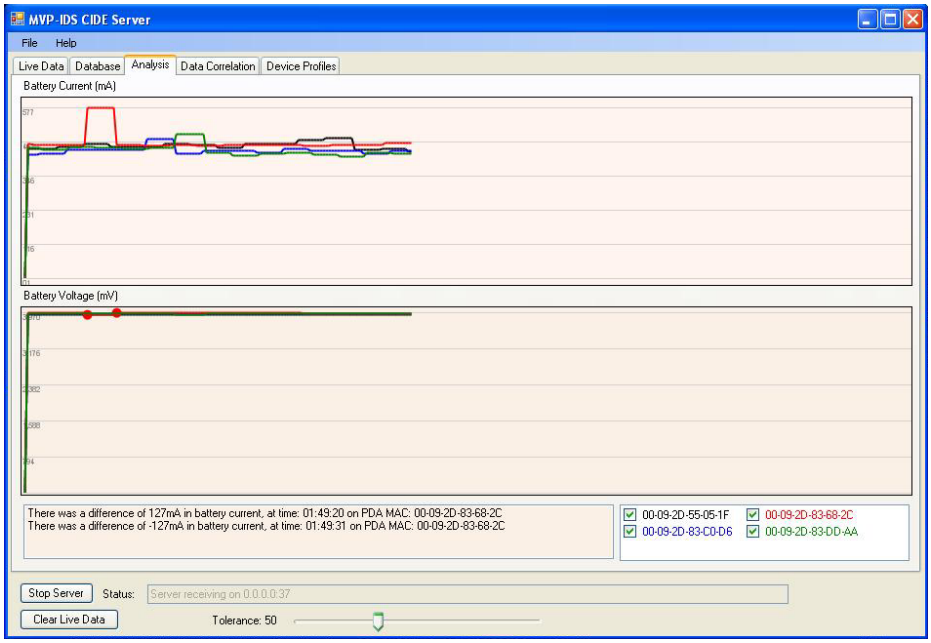


Figure 30. MVP-IDS CIDE server Analysis View.

4.4.4 Data Correlation View

The Data Correlation View, shown in Figure 31, is a graphical display of the methodology in this research: identifying IC anomalies from B-SIPS clients and validating those reports with real-time Wi-Fi and Bluetooth attack traffic. The major changes to this view from Buennemeyer's implementation involve the real-time correlation of IC anomalies with Bluetooth and Wi-Fi attack traffic, and the creation of an administrative response mechanism.

Buennemeyer theorized about correlating IC anomalies with Snort, but only with the development of MVP-IDS, has his theory been proven. This research has also extended Buennemeyer's theory by showing that IC anomalies can also be correlated with other attack mediums, if appropriate systems are in place to constantly monitor them.

Since the correlated attack data frame in Figure 31 is simple and generic for attacks from all different mediums, a new feature was added to the CIDE server. When a correlated attack is double clicked, a popup window is generated to display all details from the associated attack. This is shown in Figure 32.

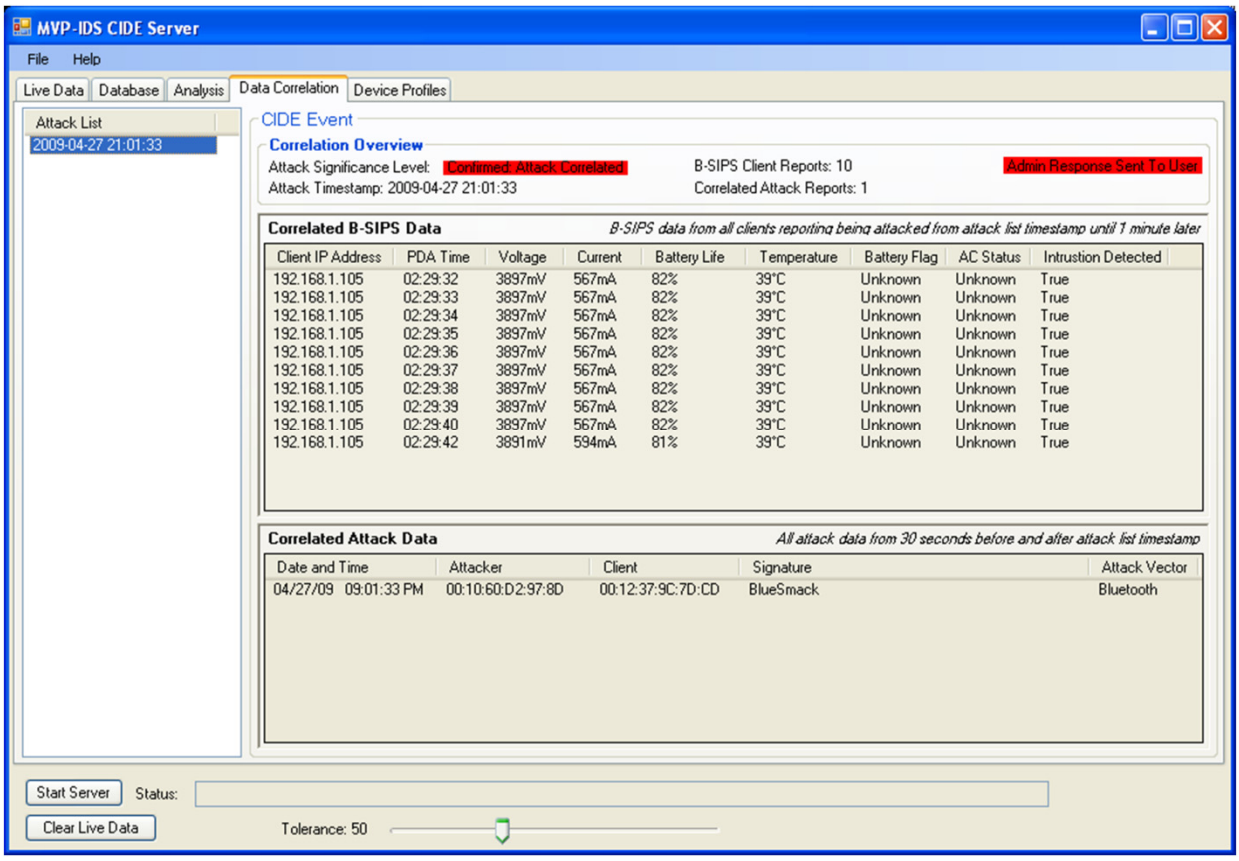


Figure 31. MVP-IDS CIDE server Data Correlation View.

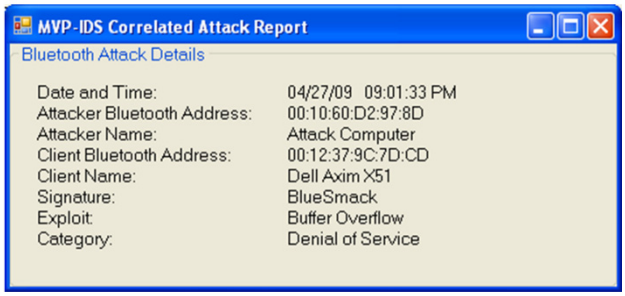


Figure 32. MVP-IDS CIDE server Data Correlation View popup window.

4.4.4.1 Real-Time Attack Correlation

The real-time attack correlation feature is a new and significant addition to MVP-IDS. It is conducted in the following six steps shown below.

1. The CIDE server continually receives B-SIPS client status reports from PID users. If a report contains an intrusion flag, indicating an IC anomaly, the CIDE server then begins conducting a real-time attack correlation with the B-SIPS client status report and attacks logged in the Snort and BADSS attack databases.
2. The CIDE server first checks the Snort database for attacks occurring in previous 30 seconds before the received B-SIPS client status report and 30 seconds after it. A 30 second window was chosen as a sufficient time interval to correlate attacks, primarily based on Buennemeyer's previous work [1] and to account for variable smart battery polling rates. Within this 60 second time interval, the CIDE server attempts to match IC anomalies with Wi-Fi attacks based on timestamps and IP addresses. If those two fields produce matching information, the CIDE server has correlated an IC anomaly with a Snort recorded attack.
3. Next, the CIDE server repeats step 2, only this time polling the BADSS attack database. In order for the CIDE server to correlate an IC anomaly with a BADSS attack record, matching timestamps and Bluetooth device addresses most occur. If these two fields produce a match, the CIDE server has correlated a Bluetooth attack.
4. This step is only executed if IC anomalies are correlated with both a Snort attack record and a BADSS attack record. When an attack is launched using multiple attack mediums, it is said to be a blended attack. This research currently implements two Bluetooth/Wi-Fi blended attacks.
    a. **BlueSYN [1]:** This attack involves attacking a device by simultaneously launching a BlueSmack *l2ping* flood and an *hping3* SYN flood. BlueSYN was developed by researchers with the intent to implement an extreme battery exhaustion DoS attack.

    b. **PingBlender [1]:** This is also another attack that was created to perform extreme battery exhaustion. Its components are an ICMP ping flood and Bluetooth *l2ping* flood.

If either of the two previously mentioned attacks is recognized, the attack is logged to a blended attack database, as shown in Figure 33. If the attack is neither of the previous two, it is logged to the database as an unknown blended attack. While the attack may be a zero day attack or just a new combination of two attacks from different mediums, it still allows an SA to further investigate the attack and determine what triggered the insertion of an attack record in the blended attack database.

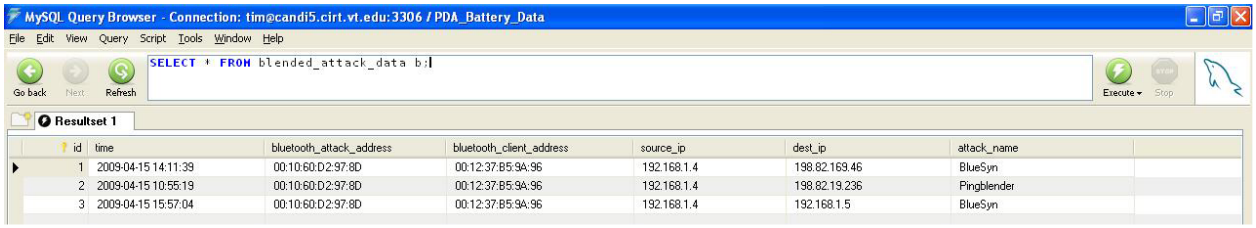


Figure 33. Blended attack database.

5. There are two possible classifications for B-SIPS client status reports which contain an intrusion flag, marking an IC anomaly:
    a. **Confirmed:** If the CIDE server has correlated an IC anomaly with either a Snort attack record, a BADSS attack record, or both, the attack is said to be confirmed.

    b. **Possible:** If the CIDE server can't match the IC anomaly with a Snort or BADSS attack record, the attack is said to be possible. Although unlikely, the attack could be one that is not in the signature database of either of the Snort or BADSS modules. Since it can't be proven with a level of confidence that the IC anomaly is neither an attack, nor a false positive trigger, the IC anomaly has to be labeled as a possible attack. By the use of this process, IC anomalies labeled as possible attacks, could be detected and a signature for the zero day attack could be entered into the appropriate signature database to allow for attack recognition on all successive attempts.

6. If the IC anomaly is labeled as a possible attack, it is recorded by the CIDE server and the process is finished. If an IC anomaly is correlated with attack records and labeled as confirmed, the CIDE server then builds an appropriate administrative response to the attack so that it can be sent to the corresponding B-SIPS client.

4.4.4.2 Administrative Response Mechanism

In Buennemeyer's version of the B-SIPS project, B-SIPS clients sent device status reports to the CIDE server, but never received any response because the system was implemented with uni-directional communication. An improvement that has been added in the creation of MVP-IDS is changing the system to operate in a bi-directional manner. Now, not only do B-SIPS clients send device status reports to the CIDE server, but the CIDE server also then replies when action needs to be taken to secure the device and alert the user of malicious activity. Responses that are sent from the CIDE server to a B-SIPS client are done in the form of an administrative response, as shown in Figure 34. They are comprised of three components and displayed to the PID user upon receipt:

- **Attack Name:** This component alerts the user to which Bluetooth, Wi-Fi, or blended attack signature was positively correlated with a reported IC anomaly trigger.
- **Attack Medium:** Attacks can be launched using Bluetooth, Wi-Fi, or a combination of the two. This component allows the user to know which wireless interface(s) the attack is originating from.
- **Administrative Action:** An administrative action is a command that will change the functionality of the Bluetooth or Wi-Fi radio on the PID. With MVP-IDS' attack medium differentiation system, only wireless interfaces deemed under attack are disabled. This is a significant improvement from Buennemeyer's implementation because an attack on one interface does not warrant disabling both wireless radios, as Buennemeyer's did. MVP-IDS only requires radios that are being used for an attack medium to be disabled.

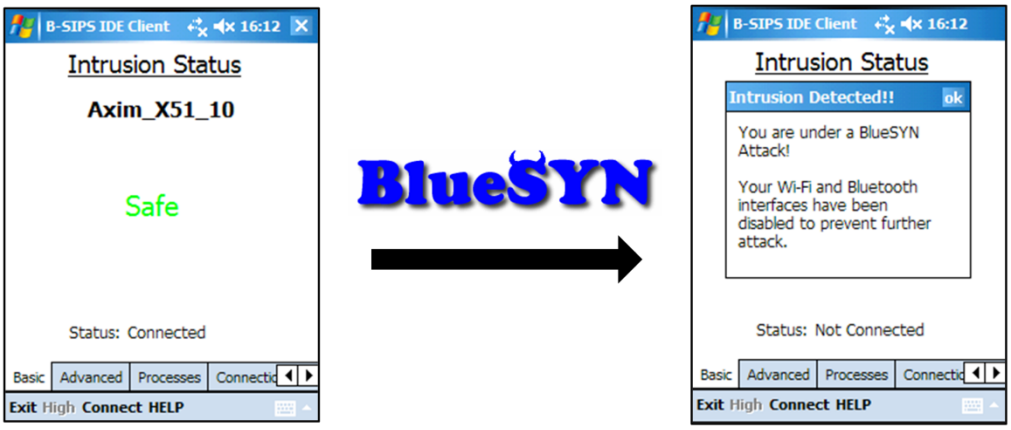


Figure 34. Administrative response to a BlueSYN attack.

4.4.5 Device Profiles View

The Device Profiles View, shown in Figure 35, is used to track statistical information on a per device basis and to, in effect, build a device profile for each B-SIPS client PID that is reporting data to the CIDE server.

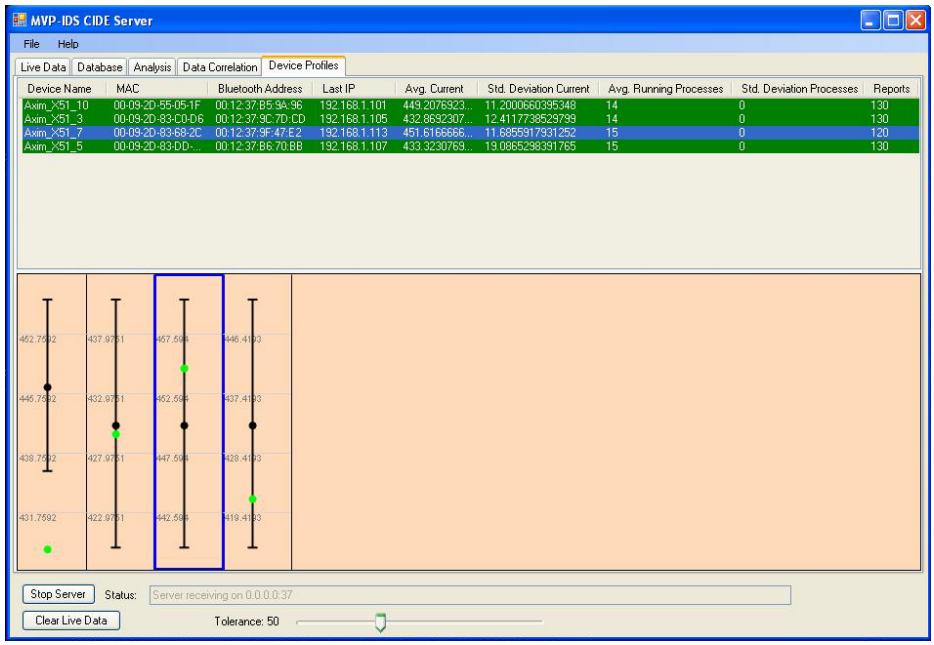


Figure 35. MVP-IDS CIDE server Device Profiles View.

While using the device profiles view, when a device listview row is double-clicked, a device profile popup window is generated listing all properties and characteristics of the selected B-SIPS PID. An example of the device profile popup window is shown in Figure 36. Only a minor change has been made to the device profiles view; the Bluetooth device address is now listed in the device listview.

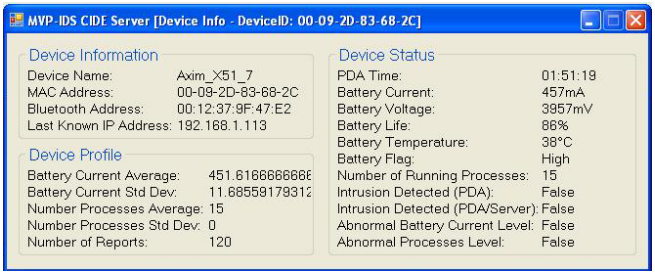


Figure 36. MVP-IDS CIDE server Device Profiles View popup window.

5. MVP-IDS Testing and Results

The main scope of this research focuses on being able to recognize attacks, but to do so, in order to prolong battery life. Attacks were developed and deployed to gain insight on MVP-IDS' effectiveness and efficiency. Section 5.1 discusses the test-bed setup, including: devices, test trial setup, attack tool use, and data collection techniques. Section 5.2 presents the results and an initial evaluation of the system. Section 5.3 is an analysis of the results, as well as a summary of knowledge gained through the deployment and testing of MVP-IDS.

5.1 Test-Bed Setup

In order to obtain accurate and repeatable results, all tests on MVP-IDS were conducted in a closed laboratory environment. Section 5.1.1 discusses B-SIPS client deployment to PIDs. Section 5.1.2 examines the setup and usage of the Snort-based Wi-Fi Module. Section 5.1.3 investigates the organization and layout of the BADSS Module. Section 5.1.4 explores the CIDE server and databases used to catalog data from each of the different MVP-IDS modules. Section 5.1.5 presents the attack tools used for obtaining results. Section 5.1.6 details the data collection methods employed used during testing.

5.1.1 B-SIPS Client Deployment to PIDs

The B-SIPS client application was developed originally by Buennemeyer using Microsoft Visual Studio 2003 and the C# programming language. Also used to augment in the development of the B-SIPS client was the .NET Compact Framework and 32feet.Net. 32feet.Net is a software development kit, much like the .NET Compact Framework, but adds extra Bluetooth and Wi-Fi functionality for Windows Mobile programmers. Since Buennemeyer had developed a solid code-base foundation, this research simply added enhanced functionality features to the design. Microsoft Visual Studio offers a simulator when developing applications for mobile devices, but in order to test the B-SIPS client and its network communication features, the code had to be deployed to PIDs through the use of Microsoft ActiveSync.

The B-SIPS client was deployed to six Dell Axim X51 PDA's. This was done in order to not only have a set of devices to gather data with, but also to have the ability to compare data within a device set. The Dell Axim X51 PDAs each have the following specifications [1]:

- Microsoft Windows Mobile 5.0
- Intel XScale PXA270 Processor at 520 MHz
- 3.7" 640x480 color TFT VGA display
- Wi-Fi and Bluetooth wireless technologies
- 64 MB SDRAM and 128 MB Flash ROM
- Removable 1100 mAh Li-Ion Primary Battery

5.1.2 Snort Module

The Snort-based Wi-Fi Module employed a Snort sensor that monitored all Wi-Fi traffic sent and received within a subnet containing B-SIPS client PIDs. This module's hardware devices consisted of a PC running CentOS-4.0 with Snort v2.8.4.1 installed, two Linksys Wireless-g routers, a MySQL database, and a homemade network tap. They were arranged in such a fashion within a subnet to form a series of network hops that forced all traffic to be routed through the Snort sensor. For further clarification of this module, this process is shown in Figure 21.

5.1.3 BADSS Module

The BADSS Module component of MVP-IDS consisted of a PC running Microsoft XP Pro with Service Pack 3, the BADSS IDE application, a MySQL database, and the Merlin II Bluetooth protocol analyzer. The full system layout of BADSS can be seen in Figure 23. BADSS was originally intended to be an automated, stand-alone system, but due to manufacture flaws in software implementation, SA interaction is required. The automation of the BADSS module will be discussed further in the future work section of chapter 6.

5.1.4 CIDE Server and Databases

Buennemeyer's original CIDE server code base was written in Microsoft Visual Studio 2003 using the C# programming language using the .NET Framework. During the migration of the CIDE server to MVP-IDS, additional

code was added using Microsoft Visual Studio 2008. The CIDE server is an executable application that can be run on any 32-bit Windows PC with the .NET Framework installed. The CIDE server was deployed to a PC running Microsoft Windows XP Pro with Service Pack 3.

5.1.5 Attack Tools

To properly test MVP-IDS and all of its components, this research took full advantage of the many penetration testing tool kits widely available on the Internet. The main attack suite consisted of a PC running Backtrack 3 [57] configured with 2 USB Bluetooth dongles and an active connection to the subnet created by the Snort-Based Wi-Fi module.

The attack tools used for launching Wi-Fi attacks included *hping3*, *nmap, Nessus3,* and *Unicorn scan*. The Bluetooth attack tools used in this research included: *RedFang, Btscanner, BluePrint, PSM Scan, RFCOMM Scan, BlueBug, BlueSnarf, Btcrack, CarWhisperer, BlueSmack, Nasty vCard, L2CAP Header Overflow, HCIDumpCrash, Bluetooth Stack Smasher, Ping of Death, Tbear, Helomoto, Nokia N70 DoS, Tanya attacks, BlueSpam,* and *Blueper.*

5.1.6 Data Collection

There needed to be a way to accurately monitor device lifetimes, while not requiring user interaction for the battery exhaustion trials. To do this, a time logging application was developed that appended the current time to a text file at one second intervals. The time logger could then be used to monitor device lifetimes once deployed to the PIDs. When the PID's battery resources were fully depleted, the device would shutdown, thus terminating the time logger application. The previous device lifetime could then be easily obtained by subtracting the first time recorded in the time log from the last time.

5.2 Tests and Results

To gauge MVP-IDS' effectiveness at both the component level and system wide, many tests were designed to scrutinize the functionality of each. First, each module had to be tested individually to ensure reliability when used in conjunction with other modules. Section 5.2.1 presents the modular attack recognition testing that was performed on the Snort, BADSS, and CIDE server modules. Section 5.2.2 investigates the battery lifetimes of a set of PIDs, both under attack and during idle conditions, to properly access the value of an IDS on mobile devices.

5.2.1 Modular Attack Recognition Testing

Modular attack recognition testing refers to the process of determining the effectiveness of each module in the MVP-IDS architecture. The better each module performs during testing, the stronger it makes MVP-IDS on a system-wide scale. Buennemeyer showed through extensive testing that the B-SIPS client was effective at recognizing IC anomalies and was not tested any further because it was not in the scope of this research effort. Section 5.2.1.1 discusses the process by which the Snort-based Wi-Fi module was tested and the results of the testing. Section 5.2.1.2 presents the scheme that was utilized to effectively determine the reliability of the BADSS module. Section 5.2.1.3 examines the CIDE server's newly added real-time attack correlation capability and its effectiveness during testing.

5.2.1.1 Snort Attack Recognition Testing

Using network-based attacks from Table 5, the goal of this test set was to determine Snort's ability to recognize malicious packet streams and create alerts that were usable by security administrators. Also examined in this module was the theory of Snort's recognition of attacks, solely based on analyzing packet headers.

To produce results for this module, all attacks were launched from an attack laptop and directed at another laptop using Wireshark [58] for traffic capture. As the attacks were deployed, Wireshark recorded them as if they were being seen from a target device's point of view. Once the attack was stopped, the target PC then created a binary dump file that could be sent to Snort for analysis. Snort then parsed the file and returned its results for attack verification. Wireshark was used to capture the packet streams because of its ability to vary the number of packet bytes recorded during each capture section. With this ability, Wireshark first analyzed the attacks with complete packet lengths. Each attack was then repeated with captures of the smallest possible 802.11 frame (68 bytes). This was done so that the capture file would only contain packet headers and exclude payload data.

This research hypothesized that since most of the attacks listed in Table 5 were based solely on the manipulation of packet header fields; the results would show no difference between full and partial packet analysis. As Table 7 shows, the prediction was only partially correct. Some attack signatures did not trigger Snort because they apparently relied on payload data analysis.

Table 7. Snort attack recognition using varied packet sizes.

#	Attack Name	Snort Signature Recognition (Packet Headers Only) (68 bytes)	Snort Signature Recognition (Entire Packets) (68-4096 bytes)
1	Ping Flood	1/1	1/1
2	ACK Flood	1/1	1/1
3	FIN Flood	2/2	2/2
4	PUSH Flood	1/1	1/1
5	RST Flood	1/1	1/1
6	SYN Flood	1/1	1/1
7	URG Flood	1/1	1/1
8	XMAS Flood	1/1	1/1
9	YMAS Flood	1/1	1/1
10	Nessus Default Scan	12/26	26/26
11	Nmap Intense Scan	5/6	6/6
12	Nmap OS Scan	4/6	6/6
13	Nmap Quick Scan	4/4	4/4
14	Unicorn Scan	0/3	3/3

5.2.1.2 BADSS Attack Recognition Testing

As with most software systems, BADSS was tested incrementally during development. As each attack signature was added to the signature database, tests were conducted to determine proper functionality. To perform these tests, Bluetooth packet captures were obtained from [49] and the Merlin II analyzer. For an attack signature to be effective, it must not only be able to recognize attacks, but also not produce too many false positives to legitimate traffic.

Once all attack signatures were added to the BADSS IDE attack signature database, a comprehensive test was performed. This involved assembling a group of Bluetooth packet capture files, all of which would be analyzed by the BADSS IDE sequentially. The group of 104 capture files contained legitimate Bluetooth communications, as well as Bluetooth traffic recorded from common attacks listed in the BADSS IDE attack signature database. As Table 8 shows, the BADSS IDE has a 100% attack detection rate, while only producing a 2.97% false positive rating.

Table 8. BADSS attack detection rates from packet capture files.

Attack	Detection Rate	False Positive Rate
RedFang	3/3	0/101
Btscanner	3/3	0/101
Tbear	3/3	0/101
BluePrint	3/3	0/101
PSM Scan	3/3	0/101
RFCOMM Scan	3/3	0/101
BlueBug	3/3	0/101
BlueSnarf	3/3	0/101
Btcrack	3/3	0/101
CarWhisperer	3/3	3/101
Helomoto	3/3	0/101
BlueSmack	3/3	0/101
Nasty vCard	3/3	0/101
L2CAP Header Overflow	3/3	0/101
HCIDumpCrash	3/3	0/101
Nokia N70 DoS	3/3	0/101
Bluetooth Stack Smasher	6/6	0/101
Ping of Death	3/3	0/101
Tanya	3/3	0/101
BlueSpam	3/3	0/101
Blueper	3/3	0/101
Total	**66/66 = 100%**	**3/101 = 2.97%**

The BADSS IDE will only detect those Bluetooth attacks listed in its attack signature database, and if a new attack is discovered, an attack signature can easily be added. This was done to allow for growth and implemented by having separate modules in the BADSS IDE for each attack signature. Therefore, if a new attack signature is developed, a module is added to the code base and a call to that module would then allow the BADSS IDE to recognize the attack from that point forward. As [49] first showed and this research validates, Bluetooth attacks can be recognized by a packet-based IDS if the proper tools are used to monitor and parse Bluetooth communications.

5.2.1.3 CIDE Server Real-Time Attack Correlation Testing

One of the significant additions that MVP-IDS has incorporated into the CIDE server's functionality is the ability to correlate IC anomalies with real-time Wi-Fi and Bluetooth attack traffic. Testing this functionality was broken into three separate categories, one for each attack medium.

1. **Real-time correlation with Wi-Fi attacks:** With all Wi-Fi attacks logged to a MySQL database, when a B-SIPS client status report contained an IC anomaly, the Snort attack database was simply queried for matches. To sufficiently correlate an attack, matches were considered confirmed if the Snort attack

database contained an attack record 30 seconds before or after the IC anomaly and that the IP address of the B-SIPS client was the same as was contained in the Snort attack record. A 30 second window was chosen as an acceptable time interval because of variable smart battery polling rates [1].

2. **Real-time correlation with Bluetooth attacks:** Much in the same fashion as the real-time correlation for Wi-Fi attacks, Bluetooth attacks were successfully correlated. When the CIDE server received an IC anomaly from a B-SIPS client, it queried the Bluetooth attack database for an attack record containing a matching Bluetooth device address and timestamp corresponding to 30 seconds before or after the B-SIPS client status report.

3. **Real-time correlation with blended attacks:** The CIDE server's real-time correlation of a blended attack was merely a check to see if it had already correlated the IC anomaly with both a Wi-Fi attack and a Bluetooth attack from their respective attack databases. If it had, it then categorized the attack as blended. This form of real-time correlation was intrinsically successful based upon the success of the real-time correlation with each of the two previous routines.

5.2.2 Battery Drain Testing

The main objective of this research was to hinder outside sources from negatively influencing the usability and lifetime, per battery charge, of PIDs. Battery lifetimes of PIDs under different operating conditions had to be evaluated to determine the effectiveness of the MVP-IDS system. Buennemeyer first explored this area by examining battery lifetimes of Dell Axim X30 PDAs under idle conditions, running the B-SIPS client, and also testing the device under attack from a SYN flood. Next, he examined Dell Axim X51 PDAs to determine the most power conservative B-SIPS client status reporting rate. While these tests provided an initial starting point for examining the effectiveness of the B-SIPS client and PID battery lifetimes, there were still many other operating conditions that were not investigated. Rapid battery depletion due to Bluetooth, Wi-Fi, and blended attacks are the main focus of this research, mainly because these are vectors which are most vulnerable on PIDs. This research further explores these venues and tries to reinforce the idea that the B-SIPS client not only protects PIDs from wireless attacks, but also protects their associated battery lifetimes.

All devices were fully charged and set to the maximum performance state, meaning that the backlight was never dimmed, the processor was at maximum operating speed, and both the Wi-Fi and Bluetooth radios active during each trial. The time logger application was deployed to each PID to record the battery lifetime for each device in each trial. Section 5.2.2.1 established a battery lifetime of each PID under idle conditions. Section 5.2.2.2 then produced battery lifetimes of PIDs under idle conditions, but with the MVP-IDS version of the B-SIPS client running. Section 5.2.2.3 examines the effect of Wi-Fi attacks on the battery lifetime of each PID. Section 5.2.2.4

explores the battery lifetimes for PIDs while under Bluetooth attacks. Section 5.2.2.5 investigates the battery lifetime implications of blended attacks directed PIDs.

5.2.2.1 Battery Drain of PIDs Under Idle Conditions

Each PID was tested under idle conditions to determine a baseline value representing its optimum battery lifetime for comparison in later battery drain trials. Once each device was fully charged and appropriately configured to its maximum performance state, a trial was started. The time logger application was allowed to run for the duration of the battery drain trial so that when a PID was fully discharged, a total battery lifetime could be recorded. This process was repeated for 15 trials, using 6 different Dell Axim X51 PDAs. Two predictions were made regarding the results.

1. Each PID should produce its own consistent data set with very little deviation. The data set for each device should show a normal distribution with actual time trials clustering around the mean battery lifetime for each device set.

2. The battery lifetimes for each device should vary only slightly from device to device. This means that the difference between battery lifetimes sets for each device should not be statistically significant within a 95% confidence interval.

Table 9. Battery lifetime trials under idle conditions.

Dell Axim X51 Battery Lifetime Observation Under Idle Conditions (in Sec.)						
	PDA 1	**PDA 2**	**PDA 3**	**PDA 4**	**PDA 5**	**PDA 6**
Trial 1	6075	7394	6077	6583	6662	6363
Trial 2	6223	7534	6255	6989	6695	6387
Trial 3	6447	7332	6464	6828	6792	6197
Trial 4	6159	7356	6212	6686	6781	6201
Trial 5	6268	7183	6121	6712	6783	6160
Trail 6	6321	7397	6193	6774	6632	6111
Trial 7	6343	7259	6164	6913	6982	6321
Trial 8	6129	7374	6163	6810	6655	6118
Trial 9	6257	7143	6208	6698	6784	6183
Trial 10	6304	7219	6182	6739	6801	6222
Trial 11	6273	7246	6129	6673	6697	6253
Trial 12	6250	7400	6301	6705	6614	6070
Trial 13	6218	7464	6235	6551	6712	6171
Trial 14	6195	7361	6207	6810	6754	6138
Trial 15	6232	7487	6354	6667	6884	6221
Mean	6246	7343	6218	6743	6749	6208
StDev	91	113	97	116	98	91

Table 9 shows the data obtained in this test set. JMP[59] is a software application that provides many statistical analysis techniques and was chosen as the analysis tool to be used on the recorded data. To determine the validity of the predictions made before testing, two different statistical methods were used. Prediction 1 was analyzed by graphing each device's set of battery lifetimes on a normal quantile plot. As Figure 37 shows by the diagonal trend lines, each device successfully conforms to a normal distribution. Also, the standard deviation for each set of device trials is very low, all under 2 minutes. For the scope of this research, two minutes added or subtracted to a PID's battery lifetime is not significant. Therefore, the mean value from each set of trials can be used as a representative number for approximating that device's battery lifetime.

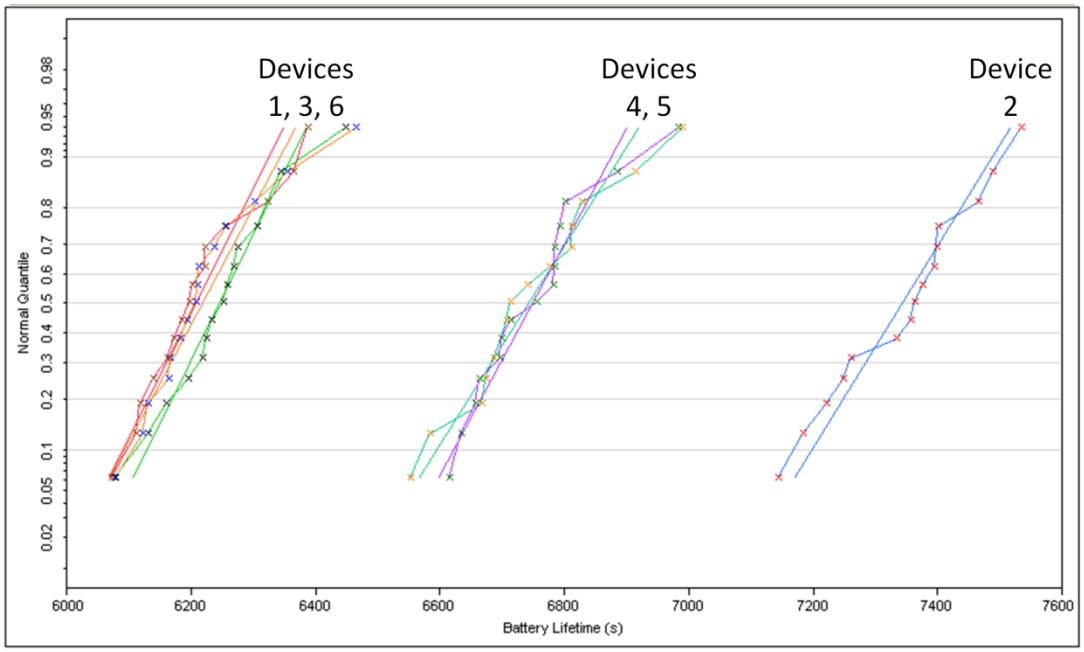


Figure 37. Normal quantile plot of PID battery lifetimes.

A One-way Analysis and Student's t-test was performed using a 95% confidence interval in order to assess prediction 2. The results of this test are shown in Figure 38, which represent statistically similar time trial sets as overlapping circles. Contradictory to prediction 2, two physically identical devices from the same manufacturer will not always produce battery lifetimes that are statistically similar. While this is a surprising conclusion to prediction 2, there is a possible explanation. Buennemeyer noted that through battery drain trending [1], PID batteries continually lose their charge capabilities on successive charge/discharge cycles. A PID that has a larger battery lifetime could simply be indicating that the device is newer than devices it is being compared to.

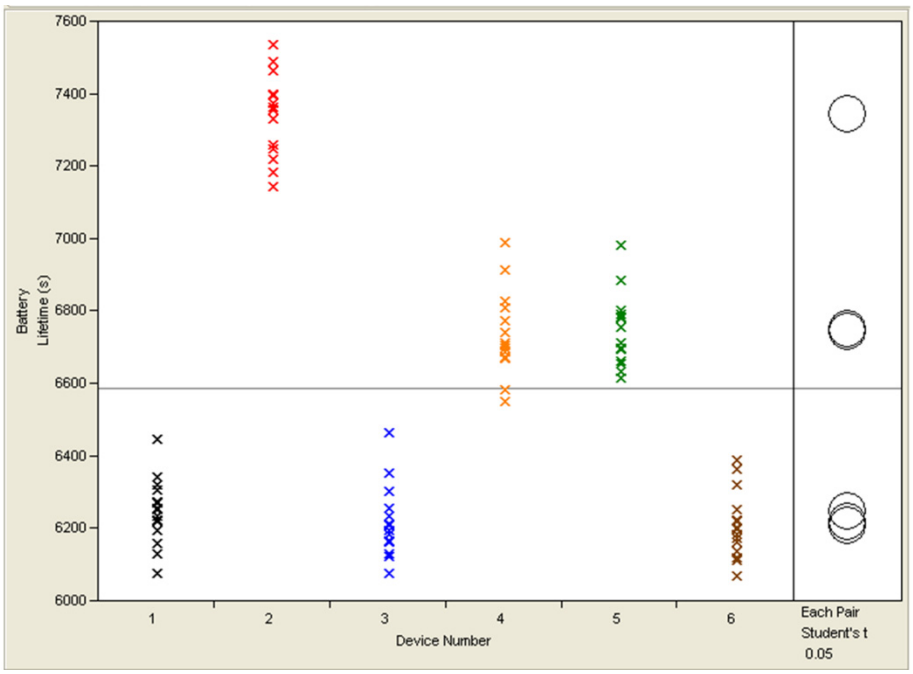


Figure 38. One-way analysis and Students t-test of PID battery lifetimes under idle conditions.

5.2.2.2 Battery Drain of PIDs Running the MVP-IDS Version of the B-SIPS Client

Previously, Buennemeyer had obtained battery lifetimes for the original B-SIPS client with a 10 second reporting rate. These values could not be used in this research effort, since the lifetimes did not take into account the time logger application or the new code changes that have been made to support the bi-directional communication mechanism of MVP-IDS. Because these modifications could greatly affect battery lifetimes, new testing had to be performed.

PID battery depletion lifetimes under idle conditions were established as a baseline to compare all other successive tests to. With this benchmark in place, the B-SIPS client was then tested for efficiency. The B-SIPS client must not have a significant negative impact on a PID's battery lifetime for it to be successful in the mobile environment. The test setup used for this set of time trials is similar to that used to obtain battery lifetimes of PIDs under idle conditions. All devices were again fully charged, configured into their maximum performance states, and timed using the time logger application. The MVP-IDS version of the B-SIPS client was started and allowed to continually run for the entire duration of the test. The recorded battery lifetimes obtained from the time logger application are shown in Table 10.

Table 10. Battery lifetime trials running the MVP-IDS version of the B-SIPS client.

Dell Axim X51 Battery Lifetime Observation with MVP-IDS' B-SIPS Client Running (in Sec.)						
	PDA 1	**PDA 2**	**PDA 3**	**PDA 4**	**PDA 5**	**PDA 6**
Trial 1	6121	7173	5986	6501	6554	5934
Trial 2	6036	7157	6085	6641	6604	6078
Trial 3	6274	7273	6063	6710	6663	6196
Trial 4	6245	7208	6212	6623	6724	6192
Trial 5	6106	7323	6229	6620	6539	6152
Trail 6	5905	7135	5903	6612	6415	5982
Trial 7	5940	7248	6033	6582	6533	6098
Trial 8	6065	7258	6131	6508	6560	6089
Trial 9	6031	7106	6079	6685	6603	6031
Trial 10	6107	7242	6197	6691	6483	6058
Trial 11	6003	7134	6054	6356	6328	5947
Trial 12	5982	7181	6123	6687	6402	6034
Trial 13	6154	7102	6127	6541	6529	6001
Trial 14	6240	7325	6092	6621	6573	6098
Trial 15	6125	7315	6334	6665	6671	6188
Mean	6089	7212	6110	6603	6545	6072
StDev	110	78	105	94	106	86

The results of Table 10 show that PID battery lifetimes are less when running the MVP-IDS version of the B-SIPS client, but only approximately 2.2% less than that of the device alone operating under idle conditions. Table 11 shows the battery lifetime differences between the means of idle PID trials and those running the B-SIPS client. Also noticed in this series of tests was that battery lifetime of PIDs running the MVP-IDS version of the B-SIPS client also conform to a normal distribution, as shown in Figure 39.

Table 11. Battery lifetime comparisons of idle PIDs vs. those running the MVP-IDS version of the B-SIPS client.

Dell Axim X51 Battery Lifetime Comparisons (in Sec.)						
	PDA 1	**PDA 2**	**PDA 3**	**PDA 4**	**PDA 5**	**PDA 6**
Idle	6246	7343	6218	6743	6749	6208
With B-SIPS Client	6089	7212	6110	6603	6545	6072
B-SIPS Client Battery Consumption %	2.51	1.78	1.74	2.08	3.02	2.19

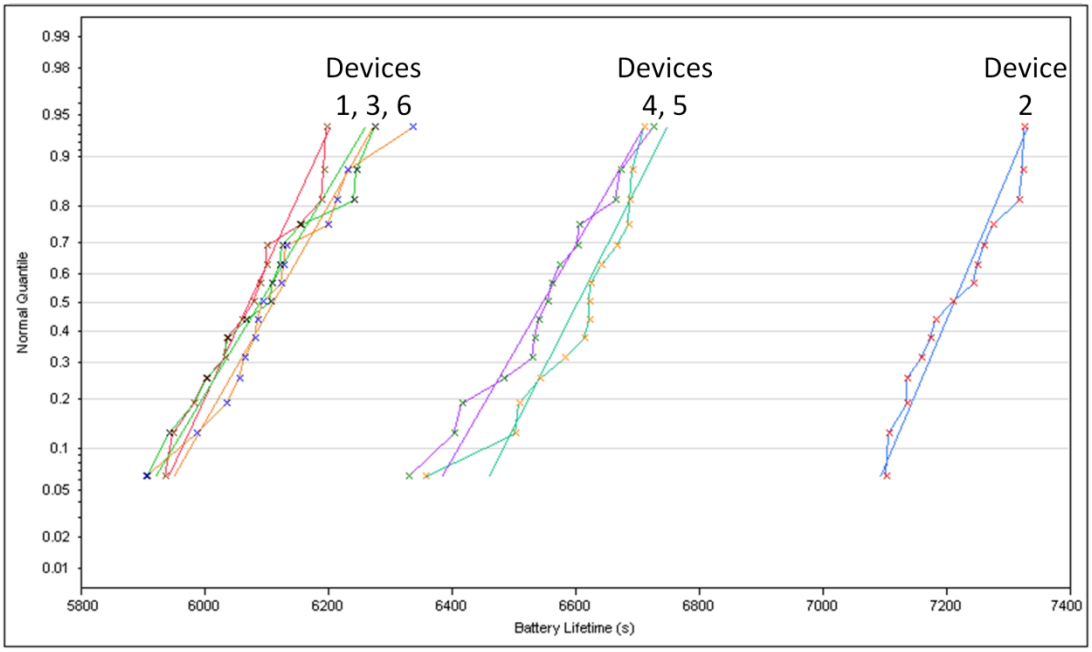


Figure 39. Normal quantile plot of battery lifetimes for PIDs running the MVP-IDS version of the B-SIPS client.

5.2.2.3 Battery Drain of PIDs Due to Wi-Fi Attacks

Two important details were discovered from the baseline testing of PIDs under idle conditions, as well as test results from running the MVP-IDS version of the B-SIPS client.

1. Not all PIDs tested produced similar battery lifetime means. Therefore, when analyzing results from other tests, time trials could only be compared to baseline values on a per device basis.

2. Since each PID tested produced battery lifetimes that were normally distributed for both previous time trial sets, it was decided that fewer time trials were needed to accurately convey the mean battery lifetime value for other tests. For the remainder of the battery lifetime testing, 5 trials were used to calculate a mean value approximating a PID's battery lifetime.

The focus of battery drain testing turned to battery exhaustion attacks via the Wi-Fi medium with these two details in mind. All PIDs were initially configured in the same manner as they were for idle baseline testing. The only difference between those tests and the current set is that another computer launched a particular Wi-Fi flooding attack at the idle PID. From the attacks listed in Table 12, only 3 were chosen as suitable to acquire valuable battery lifetime results. These attacks were a ping flood, an ACK flood, and a SYN flood.

Table 12. Wi-Fi attack suitability for battery lifetime testing.

#	Attack Name	Suitability For Battery Lifetime Testing
1	**Ping Flood**	**Successful and Repeatable**
2	**ACK Flood**	**Successful and Repeatable - Used For Comparison**
3	FIN Flood	Replicates ACK Flood
4	PUSH Flood	Replicates ACK Flood
5	RST Flood	Replicates ACK Flood
6	**SYN Flood**	**Successful and Repeatable - Used For Comparison**
7	URG Flood	Replicates ACK Flood
8	XMAS Flood	Replicates ACK Flood
9	YMAS Flood	Replicates ACK Flood
10	Nessus Default Scan	Not Applicable
11	Nmap Intense Scan	Crashes Wi-Fi NIC of PID
12	Nmap OS Scan	Crashes Wi-Fi NIC of PID
13	Nmap Quick Scan	Crashes Wi-Fi NIC of PID
14	Unicorn Scan	Crashes Wi-Fi NIC of PID

The ping flood was chosen as a testable attack because it is distinctive and produced repeatable full battery discharges. Attacks 2-9 in Table 12 closely resemble each other in attack usage. The only difference between these attacks is the bit flags that are set within the TCP packet header. Attacks 2-9 were successful, but only the SYN and ACK floods were chosen for battery lifetime testing. The SYN flood was chosen as an attack trial because it could be used for comparison during the testing of the BlueSYN blended attack. The ACK flood was chosen as a third attack trial to show that changing a bit, or series of bits, in a TCP packet would not greatly affect the battery lifetime of a PID. A Nessus default scan was not applicable to battery lifetime testing because it was not a flooding attack that would possibly lead to rapid battery depletion. Only attacks that had potential for battery exhaustion capabilities were considered for any battery drain tests. It was also discovered that successive nmap and unicorn scans would cause a malfunction in the PID's Wi-Fi NIC during pre-trail testing. Therefore, these attacks were also deemed unsuitable for battery lifetime testing.

The PID chosen for all attack testing purposes was based on random selection prior to device identification. Also noted during attack testing was that flooding attacks were not able to produce battery lifetime results. The flood option of *hping3* simply crashed the Wi-Fi NIC and caused it to become unresponsive. To combat this obstacle, the 3 Wi-Fi attacks were run at a much slower speed, only 100 packets per second. The three chosen Wi-Fi attacks are described below:

- **Ping Flood:** Ping is a troubleshooting network tool used to determine if a host is reachable on a given network. This tool functions by sending an ICMP echo request to the target host in hopes of receiving a return ICMP echo reply. A ping can act as a DoS flooding attack against a desired host at high speeds of

deployment. A ping flood attack was launched at PDA 1 using Backtrack 3 and the command: *hping3 --faster <PDA1 IP Address>*. The results of this attack and a comparison to baseline values are shown in Table 13.

Table 13. Ping flood battery drain observation and comparison.

Battery Drain Observation and Comparison of a Ping Flood on PDA 1 (in Sec.)					
Trial 1	5483			**Time**	**Percent**
Trial 2	5425		**IDLE**	6246	100.00
Trial 3	5485		**B-SIPS Client**	6089	97.49
Trial 4	5547		**Ping Flood**	5496	87.99
Trial 5	5542				
Mean	5496				
StDev	50				

- **ACK Flood:** An ACK flood was launched at idle PDA 2, much in the same manner as the ping flood was against PDA 1. This type of flooding attack is created by crafting a packet with the ACK bit set in the TCP header. Results are shown in Table 14 and the command used to implement the attack was: *hping3 --ack --faster <PDA2 IP Address>.*

Table 14. ACK flood battery drain observation and comparison.

Battery Drain Observation and Comparison of an ACK Flood on PDA 2 (in Sec.)					
Trial 1	6606			**Time**	**Percent**
Trial 2	6538		**IDLE**	7343	100.00
Trial 3	6563		**B-SIPS Client**	7212	98.22
Trial 4	6510		**ACK Flood**	6564	89.39
Trial 5	6602				
Mean	6564				
StDev	41				

- **SYN Flood:** A SYN Flood is essentially the same attack as an ACK flood, but with the SYN bit set in the TCP packet header. It was launched using the same method as the previous attacks, but was directed at PDA 3 using the command: *hping3 --syn --faster <PDA3 IP Address>.* Results are shown in Table 15.

Table 15. SYN flood battery drain observation and comparison.

Battery Drain Observation and Comparison of a SYN Flood on PDA 3 (in Sec.)					
Trial 1	5424			**Time**	**Percent**
Trial 2	5435		**IDLE**	6218	100.00
Trial 3	5556		**B-SIPS Client**	6110	98.26
Trial 4	5502		**SYN Flood**	5476	88.07
Trial 5	5465				
Mean	5476				
StDev	48				

All three Wi-Fi attacks drained their target PDA batteries approximately 10% more than during idle conditions. ACK and SYN flood attacks produced similar battery drain results, as originally predicted. The one percent difference in device battery lifetime could be attributed to the attack being directed at different PDAs.

5.2.2.4 Battery Drain of PIDs Due to Bluetooth Attacks

Battery drain testing for Bluetooth attacks was designed with a very similar setup to that of the battery drain testing for Wi-Fi attacks. First, the Bluetooth attacks from the BADSS attack signature database were analyzed to distinguish which attacks were most suitable for battery drain testing. The chosen attacks were Ping of Death, BlueSmack, BlueSpam, and Blueper floods. These attacks produced successful and repeatable results when directed at the Dell Axim X51's. Bluetooth attacks that were deemed unfit for battery drain testing were characterized as so because they either crashed the Bluetooth stack on the device or were not flooding attacks that would rapidly deplete PID batteries. Table 16 shows all Bluetooth attacks and there suitability for battery drain testing.

Table 16. Bluetooth attack suitability determination.

Attack	Suitability For Testing
RedFang	Not Applicable
Btscanner	Not Applicable
Tbear	Not Applicable
BluePrint	Repeated Trials Crash the Bluetooth Stack
PSM Scan	Repeated Trials Crash the Bluetooth Stack
RFCOMM Scan	Repeated Trials Crash the Bluetooth Stack
BlueBug	Not Applicable
BlueSnarf	Not Applicable
Btcrack	Not Applicable
CarWhisperer	Not Applicable
Helomoto	Not Applicable
BlueSmack	**Successful and Repeatable**
Nasty vCard	Not Applicable
L2CAP Header Overflow	Not Applicable
HCIDumpCrash	Not Applicable
Nokia N70 DoS	Not Applicable
Bluetooth Stack Smasher	First Attack Attempt Crashes Bluetooth Stack
Ping of Death	**Successful and Repeatable**
Tanya	Repeated Trials Crash the Bluetooth Stack
BlueSpam	**Successful and Repeatable**
Blueper	**Successful and Repeatable**

The four chosen attacks from the Bluetooth attack suitability characterization described in Table 16 were tested for their effect on PID battery lifetimes. Descriptions of these attacks are listed below:

- **Ping of Death Flood:** Much like a Wi-Fi ping, Bluetooth also has a tool, *l2ping*, to determine if a host is reachable. Using *l2ping* at a high rate of speed essentially has the same DoS effect on a Bluetooth connection as it does on a network connection. PDA5 was selected as the device to be tested and was attacked using the following command: *l2ping -f <PDA5 Bluetooth Device Address>.* Results are shown in Table 17.

Table 17. Ping of Death flood battery drain observation and comparison.

Battery Drain Observation and Comparison of a Ping of Death Flood on PDA 5 (in Sec.)					
Trial 1	5971			**Time**	**Percent**
Trial 2	5987		**IDLE**	6749	100.00
Trial 3	5980		**B-SIPS Client**	6545	96.98
Trial 4	5902		**Ping of Death Flood**	5980	88.61
Trial 5	6062				
Mean	5980				
StDev	57				

- **BlueSmack Flood:** This Bluetooth flooding attack is essentially a Ping of Death attack, but is deployed with a much larger data payload, 600 bytes. Using the 600 byte payload size sometimes causes Bluetooth stacks to malfunction on some devices, but was a successful and repeatable attack against the Dell Axim X51 PDAs. The results for this test set are shown in Table 18 and the attack was launched at PDA6 by executing the command: *l2ping -s 600 -f <PDA6 Bluetooth Device Address>*.

Table 18. BlueSmack flood battery drain observation and comparison.

Battery Drain Observation and Comparison of a BlueSmack on PDA 6 (in Sec.)					
Trial 1	5725			**Time**	**Percent**
Trial 2	5670		**IDLE**	6208	100.00
Trial 3	5503		**B-SIPS Client**	6012	96.84
Trial 4	5752		**BlueSmack Flood**	5687	91.61
Trial 5	5786				
Mean	5687				
StDev	111				

- **BlueSpam Flood:** BlueSpam, modified by this research to create vCardBlaster, is an attack that identifies Bluetooth-enabled devices in discoverable mode and spams selected targets with repeated vCard messages. This attack is most often used as an annoyance, but can be classified as a DoS flood if the rate at which the sending of the vCard messages is extremely elevated. vCards were sent as fast as possible to simulate a DoS flooding attack with hopes of depleting a PID's battery very rapidly. Results are shown in

Table 19 and the attack was launched at PDA1 using the command: *vcblaster -t 1000 -g <PDA1 Bluetooth Device Address>.* The -t option is the number of times to send a vCard and the -g option tells the program to generate a random vCard.

Table 19. BlueSpam flood battery drain observation and comparison.

Battery Drain Observation and Comparison of a BlueSpam Flood on PDA 6 (in Sec.)					
Trial 1	5194			**Time**	**Percent**
Trial 2	5043		**IDLE**	6208	100.00
Trial 3	5201		**B-SIPS Client**	6012	96.84
Trial 4	5238		**BlueSpam Flood**	5201	83.78
Trial 5	5331				
Mean	5201				
StDev	104				

- **Blueper Flood:** Designed especially for this research effort, this attack resembles BlueSpam in nature, but repeatedly floods a device with file transfers instead of vCard messages. The attack was deployed against PDA1 using the command: *blueper -i 1000 -s 1000 -e -t <file name> <PDA1 Bluetooth Device Address>.* The *-i 1000 -s 1000* in used to specify 1000 iterations of files with size 1000 byte. The *–t <filename>* specifies the file to be sent and the *–e* option adds a counter to the end of the filename so that files have unique names. Results are shown in Table 20.

Table 20. Blueper flood battery drain observation and comparison.

Battery Drain Observation and Comparison of a Blueper Flood on PDA 1 (in Sec.)					
Trial 1	5204			**Time**	**Percent**
Trial 2	5271		**IDLE**	6246	100.00
Trial 3	5130		**B-SIPS Client**	6089	97.49
Trial 4	5171		**Blueper**	5154	82.52
Trial 5	4993				
Mean	5154				
StDev	104				

Blueper and BlueSpam were surprisingly successful at draining PID battery sources based on Bluetooth's low power consumption design. Also interesting was the difference in battery drain lifetimes of the Ping of Death and BlueSmack floods. Battery lifetimes of these attacks had a difference of 3% by only changing the packet payload size. Overall, it has been shown that Bluetooth can be an effective medium in which to deploy DoS flooding attacks at battery-powered PIDs.

5.2.2.5 Battery Drain of PIDs Due to Blended Attacks

Blended attacks are those that combine two attack mediums, Wi-Fi and Bluetooth, into a single more powerful attack. In most cases, these attacks are designed with the intention of inflicting far quicker damage to a target device than is possible using only a single attack medium. It was hypothesized for this section that blended attacks would have a much larger negative impact on PID battery lifetimes than single vector attacks.

Currently, there are only two known blended attacks: BlueSYN and PingBlender. These attacks were developed during Buennemeyer's research and appeared as an excellent avenue to explore through battery drain testing in this research. It was learned that blended attacks, do in fact, put greater strain on a device than single vector attacks once testing began. Almost all battery drain time trials were not able to complete because of this. Blended attacks caused the Wi-Fi NICs to become unresponsive to any network communications. To combat this and obtain results, battery drain was monitored on a PID until the Wi-Fi NIC became unresponsive. The time the device became unresponsive was recorded and linearly extrapolated to predict a full battery drain. BlueSYN and PingBlender are described below:

- **BlueSYN Flood:** This attack involves attacking a device by simultaneously launching a BlueSmack *l2ping* flood and an *hping3* SYN flood. Results are shown in Table 21 and the commands used to implement the attack against PDA4 were:

  > *l2ping -s 600 -f <PDA4 Bluetooth Device Address>*

  > *hping3 --syn --faster <PDA4 IP Address>*

Table 21. BlueSYN flood battery drain observation and comparison.

Battery Drain Observation and Comparison of a BlueSYN Flood on PDA 4 (in Sec.)					
	Extrapolated	**Failure Time**		**Time**	**Percent**
Trial 1	5526	5526	**IDLE**	6743	100.00
Trial 2	5366	3989	**B-SIPS Client**	6603	97.92
Trial 3	5439	4515	**BlueSYN Flood**	5498	81.54
Trial 4	5485	3204			
Trial 5	5675	5675			
Mean	5498	4582			
StDev	115	1042			

- **PingBlender Flood:** Much in the same fashion as BlueSYN, this multi-vector attack uses *l2ping* and *hping3* in attempt to exhaust battery resources of target devices. The difference between this attack and BlueSYN, is that this attack is a combination of ping floods from both Wi-Fi and Bluetooth mediums. Results are shown in Table 22 and the commands used to deploy this attack against PDA2 were:

  > *l2ping -f <PDA2 Bluetooth Device Address>*

  > *hping3 --faster <PDA2 IP Address>*

Table 22. PingBlender flood battery drain observation and comparison.

Battery Drain Observation and Comparison of a PingBlender Flood on PDA 2 (in Sec.)					
	Extrapolated	**Failure Time**		**Time**	**Percent**
Trial 1	6142	3509	**IDLE**	7343	100.00
Trial 2	6222	3695	**B-SIPS Client**	7212	98.22
Trial 3	6302	5954	**PingBlender Flood**	6192	84.33
Trial 4	6135	6135			
Trial 5	6161	5761			
Mean	6192	5011			
StDev	70	1295			

Both PingBlender and BlueSYN were very successful battery exhaustion attacks compared to the entire tested attack set. However, the hypothesis for blended attacks in this section was contradicted. BlueSYN was the most effective battery exhaustion attack, but PingBlender didn't show quite the same success to validate the idea that blended attacks are always the most effective battery exhaustion attack method. Blueper and BlueSpam proved to be very malignant attacks, which is surprising due to the results obtained by the other two Bluetooth attacks. A possible answer to this anomaly is that Blueper and BlueSpam were successful not only because of the heavy Bluetooth traffic loads, but also because of the file saving and memory usage consumed on target PIDs.

5.3 Results Summary

Testing of MVP-IDS set out to prove the major point that mobile device battery lifetimes needed protection and could be accomplished by employing a non-conventional IDS to continually monitor a PID's operating characteristics. Through testing of the individual modules of MVP-IDS, it was shown that this research had successfully innovated a design to not only detect IC anomalies of PIDs, but also integrated attack signature detection modules to correlate those anomalies with malicious wireless traffic. The success of the real-time attack correlation by the CIDE server confirmed that when each attack detection module was assembled into a complete fully-functional IDS, mobile device security could be accomplished. Section 5.3.1 provides a summary of the

modular attack recognition testing. Section 5.3.2 summarizes the battery drain testing and comparisons used to analyze the effect of battery exhaustion attacks on PIDs.

5.3.1 Modular Attack Recognition Summary

It was necessary to fully assess the robustness of each module individually to ensure correct operation of MVP-IDS as a whole. If one module alone malfunctioned, it could compromise the effectiveness of the entire system. Section 5.3.1.1 discusses the test conclusions from the Snort Wi-Fi Module and analyzes its success in detecting network based attacks. Section 5.3.1.2 summarizes the results of the BADSS module testing. Section 5.3.1.3 reviews the testing that was done on the CIDE server during module interconnection to form a complete and fully functional IDS.

5.3.1.1 Snort-Based Wi-Fi Module Summary

The Snort-based Wi-Fi Module was evaluated using 14 different Wi-Fi attacks and showed success in recognizing attack based on entire packets. Also shown was that it was successful in attack recognition based solely upon packet header analysis. This is an important conclusion, as it validates the original idea of host-based packet capturing. When a raw socket implementation is released for Windows Mobile programming purposes, the results from this module seem to insinuate that the proposed theoretical approach would be feasible and successful.

5.3.1.2 BADSS Module Summary

The BADSS module was extremely successful at accomplishing its Bluetooth attack recognition task. The BADSS IDE provide a 100% attack detection rate, with a mere 2.97% false positive rate. These false positives arise from legitimate traffic not following the Bluetooth specification. CarWhisperer is a Bluetooth attack that circumvents the authentication process of securing a two-party connection. Some legitimate packet captures evaluated during testing did not follow the specification, and therefore was recognized as an attack by the BADASS IDE.

5.3.1.3 CIDE Server Summary

Since the Snort and BADSS Modules were proven successful, the CIDE server attack correlation was also successful. To correlate an attack, the CIDE server simply processed B-SIPS client status reports for IC anomalies and polled the attack databases for any attack matches. Therefore, if the Snort and BADSS Modules functioned correctly, an attack record was listed in the appropriate attack database for each reported IC anomaly. With all four modules working in a coherent fashion, it was shown that IC anomalies could be correlated with malicious Wi-Fi and Bluetooth attack traffic. MVP-IDS confirms that through employing a hybrid approach to intrusion detection, PIDs can be secured in malicious environments.

5.3.2 Battery Drain Testing Summary

This research has made three significant conclusions from PID battery drain testing. First, Dell Axim X51 PDA batteries drain in a normal distribution fashion, but the drain time across devices is not always statistically similar. Second, it has shown that battery exhaustion attacks should be seen as a significant threat to the field of mobile device security. The battery lifetimes of these devices need to be treated as dominant asset of the device because without a constant power source, mobile devices are crippled. Lastly, the MVP-IDS version of the B-SIPS client has shown to be effective at mitigating flooding attacks, while consuming very little battery lifetime overhead. With the B-SIPS client only use an excess of roughly 2.2% of a PID's battery lifetime, this research has shown that MVP-IDS is a viable option in securing mobile devices in malicious environments. A summary of attacks tested in this research and their associated battery drain percentage rates is shown in Table 23, as well as testing comparisons shown in Figure 40.

Table 23. Battery drain testing summary of all attacks tested.

Battery Drain Testing Summary			
Rank	**Attack**	**Wireless Medium**	**Excess Battery Drain Percentage**
1	BlueSYN	Blended	18.46
2	Blueper	Bluetooth	17.48
3	BlueSpam	Bluetooth	16.22
4	PingBlender	Blended	15.67
5	Ping Flood	Wi-Fi	12.01
6	SYN Flood	Wi-Fi	11.93
7	Ping of Death	Bluetooth	11.39
8	ACK Flood	Wi-Fi	10.61
9	BlueSmack	Bluetooth	8.39

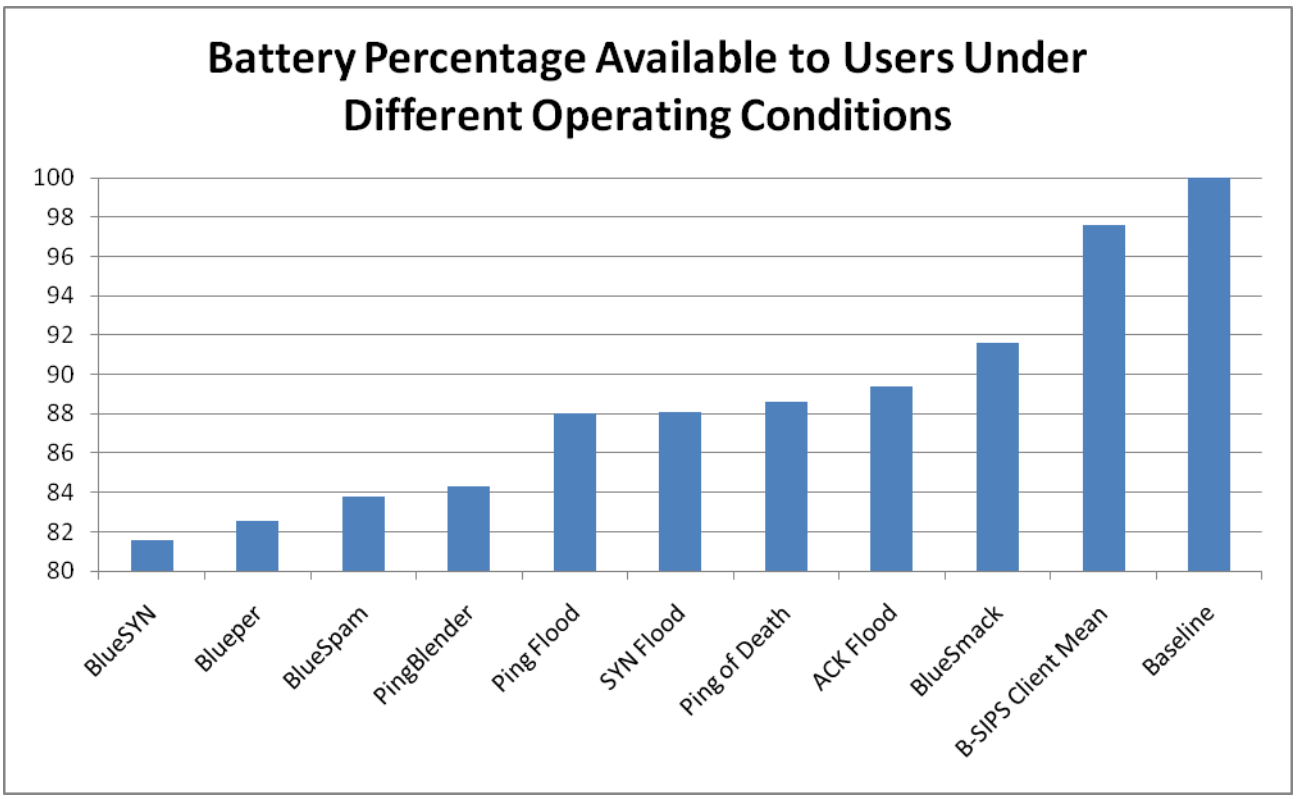


Figure 40. Battery availability to users.

6. Conclusions

MVP-IDS creates a viable solution to improve the security of PIDs. Mobile devices have an inherent need to function under stringent hardware constraints, causing the securing of these devices to often be done as an afterthought in the design process. To mitigate this design weakness and greatly enhance the security of PIDs, MVP-IDS was created. Using a hybrid approach to intrusion detection, our work confirms that PIDs can be secured in malicious environments by integrating IC anomaly triggers with attack signature correlation for Wi-Fi and Bluetooth traffic. Section 6.1 presents a summary of the research conducted throughout the design and testing of MVP-IDS. Section 6.2 recommends future work and ideas in which to further improve this research and the field of PID security. Section 6.3 provides some concluding thoughts and reflects upon this research effort.

6.1 Summary of Research

MVP-IDS is an intricate system used to correlate IC anomalies with malicious Wi-Fi and Bluetooth attack traffic. Section 6.1.1 presents a review of the MVP-IDS architecture and module design. Section 6.1.2 reiterates the significant improvements that the MVP-IDS project has added to the original B-SIPS design with which to create a more complete and robust IDS for PIDs. Section 6.1.3 provides a summary of the testing that was done on MVP-IDS and its modules to determine effectiveness and reliability.

6.1.1 MVP-IDS Review

MVP-IDS is a hybrid IDS created by the composition of four specialized and distinct attack detection modules. The B-SIPS client is the first line of defense and the triggering mechanism for MVP-IDS. It is effective by continually polling the PID smart battery for device operating characteristics such as IC, voltage, and temperature. While the B-SIPS client is monitoring each PID, the Snort and BADSS modules are passively scanning Wi-Fi and Bluetooth traffic, respectively, for communication patterns which match that of attack signatures in their attack signature databases. When a B-SIPS client recognizes an IC anomaly, it forwards it's discovery to the CIDE server in the form of a status report. The CIDE server then correlates the IC anomaly with attacks recorded by the Snort and BADSS modules. If the CIDE server determines that an IC anomaly is a confirmed match to an attack recorded by the Snort or BADSS modules, it then builds an administrative response to inform the B-SIPS client of the attack and appropriate actions that are being taken. Upon receipt of the administrative response, the B-SIPS client then disables the wireless radios that the CIDE server determined to be under attack. Wireless attacks are mitigated and PID device life is prolonged through the interaction of all four MVP-IDS modules.

6.1.2 Significant MVP-IDS Improvements

The B-SIPS project set forth a design that effectively detected IC anomalies representing attacks against a PID. Based upon its design, this research set out to innovate and extend the project to create a complete system that was more robust and better equipped to perform intrusion detection on PIDs. Based on recognized needs, the following eight significant improvements have been added to the B-SIPS project to create MVP-IDS.

1. A pattern-matching packet-based Bluetooth IDS has been added for Bluetooth attack recognition.
2. A Snort-monitored WAP scans Wi-Fi traffic and logs attacks to a CIDE-viewable database.
3. With the use of Snort for Wi-Fi traffic and BADSS for Bluetooth traffic, differentiating attacks and attack mediums is now possible.
4. Recognition of blended, or multi-vector attacks, which incorporate attacks from both Wi-Fi and Bluetooth, is now possible with MVP-IDS.
5. With Snort and BADSS logging attacks to a centralized database, the CIDE server is now able to conduct real-time correlation from B-SIPS clients as soon as they report an attack.
6. By design, the B-SIPS project only allowed for client to server communication. MVP-IDS now has the ability to actively respond to attacks using bi-directional communication.
7. When an attack was detected by the original B-SIPS implementation, both the Wi-Fi and Bluetooth interfaces were disabled to mitigate attacks. With the MVP-IDS attack medium differentiation system, only wireless interfaces deemed under attack are disabled.
8. To mitigate tampering of data transactions between B-SIPS clients and the CIDE server, confidentiality, authentication, message integrity, and non-repudiation have been integrated into all message passing.

6.1.3 MVP-IDS Testing Summary

MVP-IDS was tested in a multitude of different ways to try to produce a hardened, complete, and robust system capable of acting as an IDS for PIDs. First, each module in MVP-IDS was tested for correct operation. Buennemeyer thoroughly tested the B-SIPS client and proved it successful in detecting IC anomalies. The Snort-Based Wi-Fi Module was effective at recognizing attacks, not only when analyzing complete packets, but also when detecting attacks based on packet headers alone. Using packet capture files from O'Connor [49] and the Merlin II

Bluetooth protocol analyzer, the BADSS IDE was tested to ensure Bluetooth attack detection. Upon recording a 100% detection rate, while only producing a false positive rate of 2.97%, it was decided that the BADSS module was also successful. The CIDE server had been tested by Buennemeyer, but many additions had been made to enhance its functionality, and therefore had to again be fully inspected. The CIDE server's real-time attack correlation was inherently successful because of the success in the Snort and BADSS Modules. With all four modules integrated into a complete system, MVP-IDS was shown to be effective at performing intrusion detection for PIDs.

Second, battery drain testing of 6 Dell Axim PDAs was conducted to quantify the efficiency of the MVP-IDS system and to gain insight into the impact that battery exhaustion attacks could have on unprotected PIDs. From this testing, it was discovered that the B-SIPS client application used an excess of approximately 2.2% of a PID's battery lifetime. However, if the B-SIPS client was allowed to run in the background during a BlueSYN flood attack, the B-SIPS client could mitigate the attack and preserve as much as 16% of a PIDs battery lifetime as compared with an unprotected PID. A summary of these statistics is shown in Figure 41.

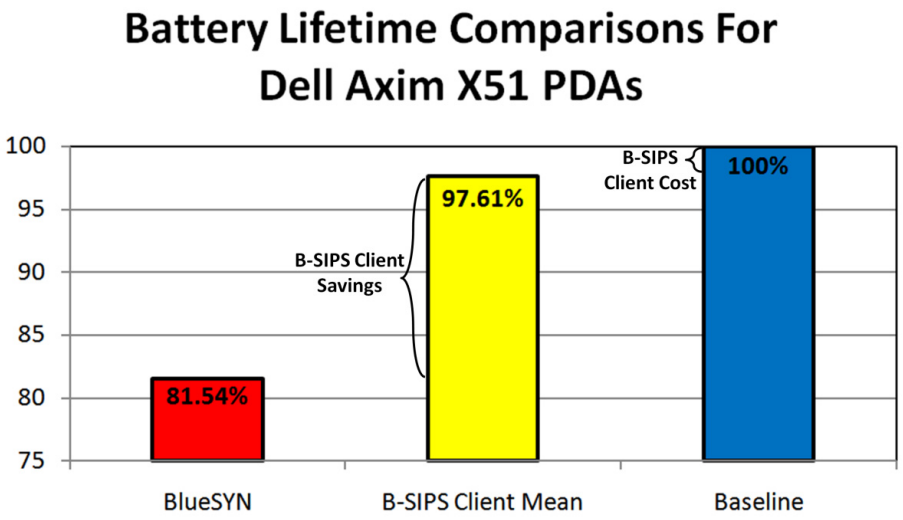


Figure 41. Effects of B-SIPS client and a BlueSYN attack on battery lifetime.

6.2 Future Work

While MVP-IDS has made great strides in PID security, there are still directions that could be taken to further this research. First, the lack of raw sockets in the Windows Mobile environment has greatly affected the design of this system. A future project could implement a driver-level implementation of the pcap library to allow access to raw socket transmissions for Wi-Fi and Bluetooth, alike. This would eliminate the need for outside wireless traffic monitoring sources, such as the Snort and BADSS modules, and also test the feasibility of this research's theoretical approach to packet capturing.

Second, MVP-IDS has only been tested with small device sets. In a real world implementation, large scale testing might be warranted to see how the system responds to increased workloads and higher bandwidth consumptions. This type of research would provide answers to feasibility uncertainties of MVP-IDS on a small or large business scale.

Third, MVP-IDS is currently platform dependent. At this time, the B-SIPS client will only run on Windows Mobile devices. Future research might examine how the principles used in creating the B-SIPS client could be ported to other platforms such as the iPhone, Android, Symbian OS. A project that created a version of the B-SIPS client that could be deployed to any PID, independent of the OS manufacturer, would also be an avenue for future research.

Lastly, a major obstacle encountered in this research was the automation of the Merlin II hardware/software package. LeCroy, the manufacturer of the Merlin II, recently deployed its product and admitted that there were manufacturer flaws in the software package that accompanied it. With user interaction, the BADSS module was effective, but could be greatly enhanced if the entire module was an automated, stand-alone system. At this point in time, LeCroy has not yet provided a solution to remedy the automation problem, but once a solution is made available, it would be appealing to test BADSS in a completely automated setting.

6.3 Concluding Thoughts

MVP-IDS was an extension of the B-SIPS project and innovated its design by creating a hybrid IDS to further explore and improve the area of PID security. The primary objective and methodology of this research effort was:

- **Objective:** To hinder outside sources from negatively influencing the usability and lifetime, per battery charge, of PIDs.
- **Methodology:** To recognize a significant change in IC on a PID and correlate the change with malicious Wi-Fi and/or Bluetooth traffic.

Both of these criterions were met and shown to be effective through the design and testing processes implemented in this research effort. In closing, it is hoped that the advances made by this research in the areas of Bluetooth security and mobile intrusion detection are not only helpful to others, but can be used to spark new and creative ideas for other researchers.

Bibliography

[1] T.K. Buennemeyer, "Battery-Sensing Intrusion Protection System (B-SIPS)," Doctoral Dissertation, Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg,VA, 2008.

[2] Snort, http://www.snort.org/, 2008.

[3] Thomas S. Heydt-Benjamin Daniel Halperin, Kevin Fu, Tadayoshi Kohno, William H. Maisel, "Security and Privacy for Implantable Medical Devices," http://www.secure-medicine.org/PervasiveIMDSecurity.pdf, 2008.

[4] Mike Roberts and Daniel Beaumont, "Bluetooth Brings Mobility to Health Care," *Planet Wireless,* pp. 11-15, 2002.

[5] Larry Shaughnessy, "Double Amputee Walks Again Due to Bluetooth," http://www.cnn.com/2008/TECH/01/25/bluetooth.legs/index.html, 2008.

[6] Barnaby J. Feder, "A Heart Device is Found Vulnerable to Hacker Attacks," http://www.nytimes.com/2008/03/12/business/12heart-web.html, 2008.

[7] Tom Paulson, "Hackers can attack heart devices," http://seattlepi.nwsource.com/local/354617_defibhack12.html, 2008.

[8] Declan McCullah, "Obama's new BlackBerry: The NSA's secure PDA?," http://news.zdnet.com/2100-9595_22-262060.html, 2009.

[9] John McHale, "Wireless devices link soldiers on the digital battlefield," http://mae.pennnet.com/display_article/89485/32/ARTCL/none/none/1/Wireless-devices-link-soldiers-on-the-digital-battlefield/, 2001.

[10] International Online Defense Magazine, "Secure PDA Phone," http://defense-update.com/products/s/secure-PDAP.htm, 2004.

[11] F. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues For Ubiquitous Computing," *Computer,* vol. 35, pp. 22-26, 2002.

[12] T. Martin, M. Hsiao, Ha Dong, and J. Krishnaswami, "Denial-of-service attacks on battery-powered mobile computers," in *Pervasive Computing and Communications (PerCom '04)*, pp. 309-318, 2004.

[13] MSDN, "TCP/IP Raw Sockets," http://msdn.microsoft.com/en-us/library/ms740548.aspx, 2008.

[14] LeCroy, "Merlin II Analyzers," http://www.lecroy.com/tm/products/ProtocolAnalyzers/MerlinII.asp?menuid=60, 2008.

[15] Craig Freudenrich Ph.D. and Carmen Carmack, "How PDAs Work," http://electronics.howstuffworks.com/gadgets/travel/pda.htm/printable, 2009.

[16] Mobile Tech Review, "What is a PDA?," http://www.mobiletechreview.com/genfaq.shtml, 2009.

[17] Jo Best, "Analysis: What is a smart phone?," http://networks.silicon.com/mobile/0,39024870,39156391,00.htm, 2006.

[18] Liane Cassavoy, "What Makes a Smartphone Smart?," http://smartphones.about.com/od/smartphonebasics/a/what_is_smart.htm, 2009.

[19] David Needle, "Smartphones Take Center Stage," http://www.wi-fiplanet.com/news/article.php/3551686, 2005.

[20] Gartner, "Gartner Says Worldwide Smartphone Sales Reached Its Lowest Growth Rate With 3.7 Per Cent Increase in Fourth Quarter of 2008," http://www.gartner.com/it/page.jsp?id=910112, 2009.

[21] Michelle Megna, "2009 Smartphone Sales: Dipping Sharply," http://itmanagement.earthweb.com/cnews/article.php/3810521/2009%20Smartphone%20Sales:%20Dipping%20Sharply.htm, 2009.

[22] William Stallings, Cryptography and Network Security: Principles and Practices, 4 ed. Upper Saddle River, NJ: Prentice Hall, 2006.

[23] Microsoft MSDN, "Data Confidentiality," http://msdn.microsoft.com/en-us/library/aa480570.aspx, 2009.

[24] Ed Skoudis with Tom Liston, Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses. Upper Saddle River, NJ: Pearson Education, Inc., 2006.

[25] Andrew Tanenbaum, Computer Networks, Second ed. Englewood Cliffs, NJ: Prentice Hall, 1988.

[26] Douglas Comer, Internetworking with TCP/IP vol. I: Principles, Protocols, and Architecture. Upper Saddle River, NJ: Prentice Hall, 1995.

[27] Larry Peterson and Bruce Davie, Computer Networks: A Systems Approach, 4 ed. San Francisco, CA: Morgan Kaufman Publishers, 2007.

[28] Wikipedia, "OSI Model," http://en.wikipedia.org/wiki/OSI_model, 2009.

[29] Wi-Fi Alliance, "About the Alliance," http://www.wi-fi.org/about_overview.php, 2009.

[30] H. Labiod, H. Afifi, and C. De Santis, Wi-Fi, Bluetooth, Zigbee and WiMax. The Netherlands: Springer, 2007.

[31] T. Karygiannis and L. Owens, "Wireless Network Security 802.11, Bluetooth and Handheld Devices," http://csrc.nist.gov/publications/nistpubs/800-48/NIST_SP_800-48.pdf, November 2002.

[32] J. Bray and C.F. Sturman, Bluetooth Connect Without Cables. Upper Saddle River, NJ: Prentice Hall PTR, 2001.

[33] Bluetooth SIG, "Bluetooth Specification Version 2.1 + EDR," 2007.

[34] Bluetooth Special Interest Group, "Bluetooth," http://www.bluetooth.com/bluetooth/, 2009.

[35] K. Scarfone and J. Padgette, "Guide to Bluetooth Security," http://csrc.nist.gov/publications/nistpubs/800-121/SP800-121.pdf, 2009.

[36] Dominic Spill and Andrea Bittau, "BlueSniff: Eve meets Alice and Bluetooth," http://www.usenix.org/event/woot07/tech/full_papers/spill/spill_html/, 2009.

[37] IEEE Standards Association, "IEEE OUI and Company_id Assignments," http://standards.ieee.org/regauth/oui/index.shtml, 2009.

[38] K. Hypponen and K. M. J. Haataja, "“Nino” Man-In-The-Middle Attack on Bluetooth Secure Simple Pairing," in *Internet, 2007. ICI 2007. 3rd IEEE/IFIP International Conference in Central Asia on*, pp. 1-5, 2007.

[39] Aurobindo Sundaram, "An Introduction to Intrusion Detection," *Crossroads, Special Issue on Computer Security,* vol. 2, pp. 3-7, 1996.

[40] SANS Institute, "Intrusion Detection Systems; Definition, Need and Challenges," http://www.sans.org/reading_room/whitepapers/detection/intrusion_detection_systems_definition_need_and_challenges_343?show=343.php&cat=detection, 2001.

[41] Ed Thompson, "Smart Batteries to the Rescue," http://www.mcc-us.com/SBSRescue.pdf, 2000.

[42] Microsoft, "Advanced Power Management v1.2," http://www.microsoft.com/whdc/archive/amp_12.mspx, 2008.

[43] "Advanced Configuration and Power Interface," http://www.acpi.info/, 2009.

[44] D. C. Nash, T. L. Martin, D. S. Ha, and M. S. Hsiao, "Towards an intrusion detection system for battery exhaustion attacks on mobile computing devices," in *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on*, pp. 141-145, 2005.

[45] G. A. Jacoby and N. J. Davis, "Battery-based intrusion detection," in *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, pp. 2250-2255 Vol.4, 2004.

[46] G. A. Jacoby, R. Marchany, and N. J. Iv Davis, "Battery-Based Intrusion Detection: A First Line of Defense," in *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, pp. 272-279, 2004.

[47] G. A. Jacoby, R. Marchany, and N. Davis, "How Mobile Host Batteries Can Improve Network Security," *Security & Privacy, IEEE,* vol. 4, pp. 40-49, 2006.

[48] Bluetooth SIG, "Bluetooth® Wireless Technology Surpasses One Billion Devices," http://bluetooth.com/Bluetooth/Press/SIG/BLUETOOTH_WIRELESS_TECHNOLOGY_SURPASSES_ONE_BILLION_DEVICES.htm,

[49] T.J. O'Connor, "Bluetooth Intrusion Detection," http://www.lib.ncsu.edu/theses/available/etd-03212008-135411/unrestricted/etd.pdf, 2008.

[50] MSDN, "Socket Programming," http://msdn.microsoft.com/en-us/library/ms172494.aspx, 2009.

[51] C. T. R. Hager, S. F. Midkiff, J. M. Park, and T. L. Martin, "Performance and energy efficiency of block ciphers in personal digital assistants," in *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pp. 127-136, 2005.

[52] J. Grossschadl, S. Tillich, C. Rechberger, M. Hofmann, and M. Medwed, "Energy Evaluation of Software Implementations of Block Ciphers under Memory Constraints," in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07*, pp. 1-6, 2007.

[53] Sun Microsystems Inc., "What is a Socket?," 2008.

[54] Network Sorcery, "Network Communications," http://www.networksorcery.com/, 2009.

[55] Cisco, "Ethernet," http://www.cisco.com/en/US/docs/internetworking/technology/handbook/Ethernet.html, 2009.

[56] Symantec, "Attack Signatures," http://www.symantec.com/business/security_response/attacksignatures/index.jsp, 2009.

[57] Remote-Exploit, "Backtrack 3," http://www.remote-exploit.org/backtrack_download.html, 2008.

[58] CACE Technologies, "Wireshark," 2009.

[59] JMP, http://www.jmp.com/, 2009.