Bluetooth Network-Based Misuse Detection

MAJ Terrence OConnor
NC State University
Raleigh, NC
terrence.j.oconnor@us.army.mil

Dr. Douglas Reeves
NC State University
Raleigh, NC
reeves@csc.ncsu.edu

Abstract

Bluetooth, a protocol designed to replace peripheral cables, has grown steadily over the last five years and includes a variety of applications. The Bluetooth protocol operates on a wide variety of mobile and wireless devices and is nearly ubiquitous. Several attacks exist that successfully target and exploit Bluetooth enabled devices. This paper describes the implementation of a network intrusion detection system for discovering malicious Bluetooth traffic. The work improves upon existing techniques, which only detect a limited set of attacks (based on measuring anomalies in the power levels of the Bluetooth device). The new method identifies reconnaissance, denial of service, and information theft attacks on Bluetooth enabled devices, using signatures of the attacks. Furthermore, this system includes an intrusion response component to deflect attacks in progress, based on the attack classification.

This paper presents the implementation of the Bluetooth Intrusion Detection System and demonstrates its detection, analysis, and response capabilities. The tool includes a visualization interface to facilitate the understanding of Bluetooth enabled attacks. The experimental results show that the system can significantly improve the overall security of an organization by identifying and responding to threats posed to the Bluetooth protocol.

1 Introduction

1.1 Bluetooth-Enabled Technology

The Bluetooth Special Interest Group developed the Bluetooth wireless communications protocol (IEEE 802.15.1 standard) for a multitude of mobile devices. In near ubiquity now, over 15 million Bluetooth radios shipped per week in 2007, with over 1.8 billion Bluetooth devices in existence currently.[1, 2] Examples of Bluetooth devices include smart-phones, handheld computers, hands-free audio devices, global-positioning devices, and wireless peripherals.

Bluetooth devices offer an attractive target for hackers, because physical access is not required to attack such device. Furthermore, a multitude of attacks exist that can compromise the security of Bluetooth-enabled devices. These attacks focus primarily on the 1.8 billion devices running the previous Bluetooth protocol version. This paper proposes a system for detecting malicious attacks on the Bluetooth communications protocol.

Bluetooth-enabled devices extend to many critical applications. Examples include the health care, banking, and military applications. Following are some examples of the seriousness of the threat due to attacks on the Bluetooth protocol.

1.1.1 Health Care Bluetooth-Enabled Technology

The health care industry utilizes Bluetooth technology. The Bluetooth specifications provide a generic profile for medical devices.[3] In health care environments, Bluetooth-enabled devices allow patient mobility. Bluetooth serves as an attractive protocol for health care administrators because of its low cost, low power consumption, and robustness.[4] Bluetooth-enabled devices exist for heart-rate monitors, glucometers, respirators, hearing aids, sleep monitors, and patient records.[5] Bluetooth-enabled medical devices require the highest level of security measures to protect critical and confidential applications. A denial of service attack on a floor of heart-rate monitors could overwhelm a hospital staff. Intercepting and decoding the packets of a hearing aid provides the equivalent of an audio bug. Compromising Bluetooth-enabled devices to reveal hospital records could compromise private, sensitive, and potentially embarrassing patient information. While Bluetooth certainly adds convenience for patients, vendors must ensure mobile medical devices comply with the generic Bluetooth Medical Device Profile and enforce strict security measures.[3] This reality requires medical facilities to employ a means of detecting malicious attacks on medical Bluetooth-enabled devices.

1063-9527/08 $25.00 © 2008 IEEE
DOI 10.1109/ACSAC.2008.39

1.1.2 Financial Sector Bluetooth-Enabled Technology

Financial establishments have also begun implementing mobile banking applications utilizing Bluetooth. Mobile banking includes checking account balances on a Bluetooth-enabled device, paying bills, or using a Bluetooth-enabled device to make purchases in brick-and-mortar stores.[6] According to recent research by the Celent financial advisory firm, 200,000 US households use some form of mobile banking.[6] By 2010, the market is expected to grow to 17 million US households.[6] In Mexico, BBVA Bancomer has deployed more than 13,000 Bluetooth-enabled payment terminals.[7] Mobile banking provides flexibility and convenience for consumers. However, mobile banking via Bluetooth presents a risk. A Bluetooth initiative by Bank of America resulted in failure when Air Defense Inc. security experts intercepted Bluetooth communications for a wireless fingerprint reader.[8] While no generic profile for mobile banking exists for Bluetooth, application developers must design systems with security in mind and require a protection mechanism for detecting malicious Bluetooth traffic.

1.1.3 Military Bluetooth-Enabled Technology

The military also suffers from vulnerabilities of the Bluetooth protocol. To illustrate the scope of the threat to the military, a researcher can examine a recent Naval recruiting campaign. As a method of recruiting, the Navy constructed a system that distributed motivational videos to nearby Bluetooth devices. The Navy placed the system at key locations on 13 different Naval posts. During a one-month experiment, the program discovered 11,000 unique Bluetooth mobile devices and delivered video to 2,000 devices.[9] Although benign in nature, the program provides insight into the scope of potential targets. Instead of distributing benign videos, the program could have delivered malware via the same mechanism and with the same relative ease.

Applications for Bluetooth extend to very sensitive military devices and programs. Bluetooth-enabled devices process sensitive information such as the exchange of data for the Common Access Card (CAC).[10] The CAC serves as a identification card that allows a member to access controlled facilities and services. By transmitting CAC information over Bluetooth, the military allows the potential capture and retransmission or decryption of such traffic by hostile attackers. Further, an ongoing program at the Defense Advanced Research Project Agency (DARPA) includes Bluetooth communication for the LANdroids projects.[11] As wireless robotics, LANdroids attempt to create a secure wireless mesh network in urban settings. Additionally, the Air Force Research Laboratory (AFRL) projects include a Bluetooth-connected swarm of miniature helicopters.[12] The Space and Naval Warfare Systems Center projects contain a Bluetooth-enabled mobile robot.[13] Even Bluetooth-enabled devices used for military applications prove potentially vulnerable to different Bluetooth attacks and require protection mechanisms.

2 Security Features of the Bluetooth Protocol

2.1 Pairing and Authentication Process

In order to communicate securely, Bluetooth devices require pairing. Pairing requires that devices exchange protected passkeys in order to create a linkkey used for encryption. The Simple Pairing protocol in the Core Specification 2.1 includes significant improvements including a Diffie Helman key exchange.[14] However, over 1.8 billion Bluetooth-enabled devices exist that operate pre-2.1 specifications.

In the previous specifications, each device creates an initialization key based on the Bluetooth MAC address, PIN passkey, and 128-bit random number.[3] Each device then uses the initialization key to exchange random words used in the creation of the linkeys. Following creation of the linkkeys, each device pair perform mutual authentication. Should an attacker be able to observe the pairing process, he can reconstruct the linkkeys to decrypt further traffic between paired devices.[15, 16]

In response to the discovered protocol weaknesses, the Bluetooth Special Interest Group developed Secure Simple Pairing. Simple Pairing uses the Elliptic Curve Diffie Helman public key exchange to protect against passive eavesdropping.[14] Initially, each device computes a public and a private key. However, only the public keys are transmitted over the radio. Thus, an eavesdropper only has access to the two public keys and cannot compute either the private key or the shared Diffie-Helman key. Once each device is authenticated, the key is also used as one of the variables to create the shared linkkey for encryption. In the latest Bluetooth specification, an encryption key can be recreated for communication sessions that last longer than 24 hours.

Although Secure Simple Pairing provides protection against passive eavesdropping, it provides no additional protection against the existing man-in-the-middle attacks.[17] Additionally, Secure Simple Pairing also introduces Near-Field-Communication (NFC) cooperation. By bringing two devices within a close proximity, the algorithm allows for automatic pairing.

2.2 Security Modes

While pairing provides the linkkey used for encryption and authentication, the Link Manager Protocol (LMP) di-

rects the security mode. Four modes exist for Bluetooth security.[14] In the first mode, a device does not initiate security procedures. In the second mode, a device does not initiate security procedures prior to the establishment of the L2CAP connection. In the third mode, the device must initiate security procedures prior to establishment of the LMP connection. In the fourth and final mode, the device can classify security requirements based on authentication and security required.

Device security in Bluetooth has improved with each release of the Core Specification.[17, 14, 3] But with all new releases comes the potential for newer attacks. Although security design and implementation prove important, the next section addresses some countermeasures a user can take to decrease the threat posed by Bluetooth-enabled attacks.

2.3 Security Countermeasures

The National Institute of Standards and Technology (NIST) provides a thorough overview on countermeasures to prevent Bluetooth attacks. For further reading, NIST provides the following documentation available at nist.gov.[18] NIST documents the policies an organization must establish to protect Bluetooth users from malicious attacks and increase the relative security of Bluetooth devices.

As described previously, the passkey aids in creation of the encryption key. As such, the maximum size 16-bit passkey should always be used. Default passkeys that come with devices should be modified to a sufficient length.[18] To additionally avoid passive eavesdropping attacks, users must avoid pairing devices in public places.

Because security is optional in the Bluetooth specification, users must select the highest level security modes, disable discoverable modes, turn off unnecessary services, and turn off devices when not in use.[18] Additionally, users must enable encryption on all broadcasted transmissions and use the maximum size encryption key. Application layer security should be coupled with proper Bluetooth usage. And users should frequently check vendor information for updated firmware for devices.

While countermeasures aid in protection of Bluetooth-enabled attacks, they certainly do not provide ultimate protection. The next section provides an overview of Bluetooth-enabled attacks.

3 Examples of Bluetooth-Enabled Attacks

Below are a few examples of currently-known attacks on the Bluetooth protocol.

3.1 HIDattack

The HIDattack, proposed by Mulliner, exploits Bluetooth Human Interface Devices (HID), such as mice, keyboards or joysticks.[19] This attack takes advantage of flawed HID implementations. The Bluez Linux stack prior to 2.25, the Windows XP SP2, Widcomm, and Mac OS X stacks all fail to incorporate low-level security modes in their Bluetooth HID implementations.[19] Thus, HIDattack attack either scans for a HID server or waits passively until a user searches for a HID device. In either case, it then connects to the user and appears to be a legitimate HID device. While the attack has a low probability of success, it represents a serious threat if successful.

3.2 CarWhisperer

The CarWhisperer attack targets known vulnerabilities in hands-free audio devices. The application can inject audio into, and record live audio from, a target device. Herfurt demonstrated the successful usage of the application in 2005.[20] Manufacturers often implement hands-free audio devices with default passkeys. The passkey serves as the secret parameter to create the linkkey in Bluetooth devices prior to the 2.1 core specification. In order to develop targets, the attack scans for devices that match the appropriate hands-free-audio class. For those found to match the correct class, the attack checks the MAC address to determine the default passkey provided by the manufacturer. This default passkey is then used to create an RFCOMM connection to the vulnerable device. The attack then creates a control connection, connecting to the SCO links, which carry the audio for the Bluetooth device.[20]

3.3 BlueSnarfer

In BlueSnarfing, the attacker gains access to remote data by initiating an OBEX Push.[21] The OBEX Push Profile (OPP) generally does not require authentication, so the attacker connects without knowledge of the valid passkey. The attacker then initiates an OBEX Get request for known files such as the phone book, device calendar or message list. Marcel Holtman and Adam Laurie of the Trifinite Group discovered this vulnerability in several devices in late 2003.[21] In 2004, Laurie tested the security of Bluetooth phones in Parliament. In the course of his experiment, Laurie found 46 vulnerable phones in 15 minutes.[21]

3.4 iPhone MetaSploit

Recently, Kevin Mahaffey and John Hering of Flexilis Inc. discovered a vulnerability in the Bluetooth implementation on the iPhone.[22] They successfully managed to introduce an exploit via the Service Discovery Profile (SDP).

Utilizing a specially crafted SDP message, the attacker can load a framework of tools to attack the entire operating system of the phone. Further, the attack enables access to a root shell on the iPhone device. Mahaffey and Hering also discovered that they could simplify the discovery of the Bluetooth MAC Address for the iPhone by passive capture of WiFi traffic. The MAC address captured in WiFi traffic allows calculation of the Bluetooth MAC address.

3.5 Emerging Trends

The number of Bluetooth attacks have grown steadily over the last five years. The F-Secure Corporation currently has classified 71 attacks that spread mobile malware via Bluetooth. Researchers at Virginia Tech have shown how to combine classic Internet protocol attacks such as the SYN flood with a Bluetooth distribution scheme.[23]

As the number of attacks have grown, so have the severity of attacks and the ease of implementation. Repositories of attacks exist with source code. Because Bluetooth devices are frequently managed by users that are less security conscious, these devices are more vulnerable to attacks.[18] With almost 2 billion devices in existence, Bluetooth poses a risk to most organizations.

Several computers and mobile computing devices now exist with WiFi, Cellular and Bluetooth protocol interfaces. The recent attacks on the iPhone demonstrate how hackers can combine weaknesses of each interface to exploit the overall system. Exposing a vulnerability in any of the protocol interfaces can lead to a possible vulnerability in the others. There has been significant progress made on intrusion detection for WiFi and cellular protocols.[24] This paper therefore examines how to detect and prevent intrusions on the Bluetooth interface.

4 Related Work

This paper presents a method of detecting malicious Bluetooth traffic based on misuse detection. The following discusses related work in intrusion detection, wireless and mobile threat modeling, and analysis tools.

4.1 Intrusion Detection

Anderson first described the concept of an intrusion detection system in 1980.[25] He suggested the use of audit trails to detect intrusive behavior such as unauthorized access to files. Denning then implemented the first generic intrusion detection model in 1987 that detected intrusions by monitoring the audit records of a particular system.[26] For each subject (user), the system kept an audit record of particular services such as access to the filesystem, executables, and system calls. Denning's model provided a means of detecting intrusive behavior by examining anomalies in the audit records for particular single computer systems.

In contrast to an anomaly detection model, Kumar described a means of detecting attacks using misuse detection.[27] Kumar described a model of misuse detection, which encoded attacks as well defined patterns and monitors for those specific patterns. As processing power increased, further techniques automated intrusion detection to reduce human intervention. Gosh and Schwartzbard introduced the use of artificial neural networks in order to detect novel attacks.[28] Their work also combined the concept of using both anomaly and misuse detection models to create a hybrid system.

Open source developers have created several popular IDS tools. Roesch created SNORT as a lightweight network intrusion detection tool.[29] The design of SNORT included a packet decoder, preprocessor, and detection engine. SNORT used the libpcap packet sniffer and logger to capture and decode packets. Snort provided the capability to decode and detect intrusive behavior in Ethernet, SLIP, and raw (PPP) data link-protocol packets.[29] The system contained a rules based logging and content pattern matching engine to detect a variety of attacks and probes.

Recent works have discussed the necessity for a wireless intrusion detection system. Intrusion detection in wireless networks proves challenging since wireless IDSs cannot use the same architecture as a network IDS. Lim et. al proposed a wireless IDS that detected threats on the 802.11 protocol.[24] Additionally, the system included the ability for an active response to wireless attackers.[24] In 2004, the US Army awarded a multi-million dollar contract to AirFortress to protect 802.11 networks in use by the military.[30]

4.2 Wireless and Mobile Threat Modeling

Recent works have classified the threats posed against mobile devices. Welch provided an overview and created a taxonomy of wireless threats.[31] Biermann and Cloette examined the necessity for prevention and detection of threats against mobile devices.[32] Finally, Cache and Liu provided a comprehensive source of several wireless threat models, attacks, and implementations.[33]

Several works examined particular Bluetooth attacks and demonstrated the necessity for a Bluetooth intrusion detection system. Most notably, the Trifinite Group has implemented and released details of several Bluetooth attacks.[34, 19, 20] Additionally, A. Wool provided a means of compromising the encryption scheme of Bluetooth.[15, 16] Haataja provided a comprehensive model for examining Bluetooth attacks.[35] Finally, the National Institute of Standards and Technology (NIST) documented several flaws in the security design of the Bluetooth protocol.[18]

Two recent works have addressed the modeling and detection of specific Bluetooth attacks. Yan et al. developed a model for the growth of Bluetooth worms that relied upon Markov Chains.[36] Buennemeyer et al. provided a means of detecting Bluetooth battery depletion attacks using an anomaly detection system that measured the power levels of a particular Bluetooth device.[23]

4.3 Current Related Tools

To capture and decode Bluetooth traffic, Spill and Bittau have developed a cost-effective means of recording Bluetooth traffic, utilizing the Universal Software Defined Radio.[37] Further, they have also provided the means to write firmware for Bluetooth dongles that perform custom sniffing. A custom firmware solution would prove valuable in the further development of a Bluetooth IDS. Haataja demonstrated the ability for the Lecroy Bluetooth protocol analyzer to record unencrypted and encrypted Bluetooth traffic.[35]

Combs developed Wireshark (formerly Ethereal) as a utility to capture and analyze packets.[38] Over 500 authors currently maintain the utility and it allows for the dissection and decoding of hundreds of protocols, including the 802.11 wireless protocols. It also uses the same libpcap decoding library as SNORT. The current maintainers of the tool have begun working on integrating and translating Bluetooth Host Controller Interface (HCI) packets into a libpcap format.

Tools to assess the Bluetooth security of an organization exist; an example is AirDefense BlueWatch.[39] This tool assesses the overall Bluetooth security; however, it provides no means of intrusion detection or active response. The system proposed in this paper does both, and can be integrated with existing tools to increase the overall security of an organization. The next section describes the design of the implemented Bluetooth IDS.

5 Design

5.1 Overview

Figure 1 shows the design of the system. The implemented network IDS examines decoded Bluetooth traffic to detect malicious behavior through the use of pattern matching, and a set of plug-in modules. Additionally, the IDS provides alert, visualization, and response systems based on the output of the IDS engine. Each of these key components is described in detail below. Following that is a discussion of the reconnaissance, denial of service, and information theft attack signatures needed to detect particular Bluetooth attacks.

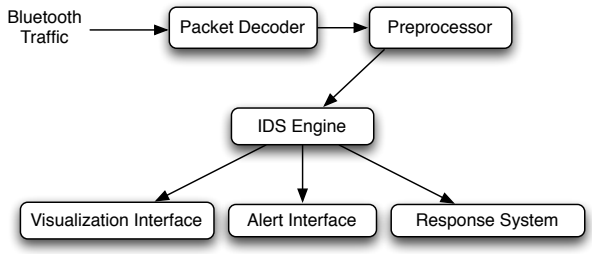


Figure 1. Design of Bluetooth Intrusion Detection System

5.2 Packet Decoding and Preprocessing

The packet decoder captures Bluetooth packets and prepares the packets for the preprocessor prior to usage by the IDS engine. The packet decoder listens to either a specific frequency, or a specific piconet, in order to capture packets. By synchronizing with the master device on a piconet, the decoder follows the hopping sequence of a particular piconet and exports decoded packets to the preprocessor.

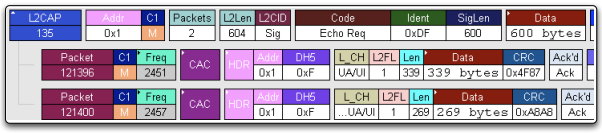


Figure 2. Packet Reassembly by the Preprocessor

The preprocessor reassembles fragmented packets, and modifies the data packets to prevent signature evasion techniques. Figure 2 depicts the reassembly of two packet fragments back into one logical packet. In addition to reassembly, the preprocessor discards several of the baseband and radio layer packets such as the Null, Polling, and Device ID packets. These packets ensure the radio and baseband layer are established but provide no helpful information to identify known attacks. By reducing these packets, the preprocessor decreases the workload of the IDS engine.

Also, the preprocessor handles reassembly and stateful inspection of streams of Bluetooth traffic, similar to the Stream4 preprocessor employed by SNORT.[29] By assembling packets into a particular Bluetooth traffic stream, the IDS engine can detect more complex attacks through the use of plug-in modules. The preprocessor uses a sliding window to manage the length of the stream exported. The system currently uses a fixed-length sliding window whose

length is determined by the time required to detect most attacks.

5.3 IDS Engine

The method includes a pattern matching engine similar to that of conventional IDS systems, such as SNORT. It has the capability to do pattern matching as well as a set of plug-in modules to analyze and detect more-complex Bluetooth attacks.[29] A grammar has been defined for specifying the rules checked by the IDS Engine.

5.3.1 Pattern Matching

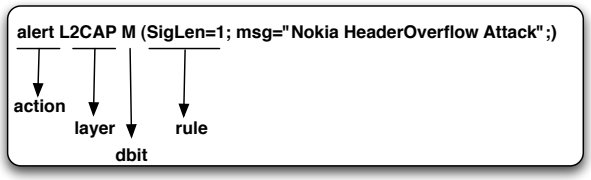


Figure 3. Rule to Detect Malicious L2CAP Header Overflow Attack

The system uses a pattern matching scheme to match malicious packets against a set of user-configurable rules. By matching signatures of known patterns of malicious traffic, the system efficiently detects attacks. Signatures can match a packet against a particular protocol layer, device, and specific fields of each packet.

For example, a signature can match the Signal Length field of an L2CAP layer packet. Figure 3 demonstrates a rule to identify a Bluetooth Header Overflow Attack. In this attack, a L2CAP layer packet originating from a Bluetooth master device has a Signal Length of 1. This Signal Length value causes some Bluetooth devices to reference an invalid memory address.

Although signatures can only detect known attacks, the user can update signatures when new types of Bluetooth attacks are discovered. The IDS provides an interface for writing new rules to detect Bluetooth attacks. Further, the rules allow the user to alert, log, or perform actions in response to a matched signature.

5.3.2 Plug-in Modules

The IDS uses plug-in modules to detect more-complex attacks that require a stateful inspection of the stream of Bluetooth traffic. Due to the fact that data is organized into a stream by the preprocessor, the plug-in modules can find attacks that use multiple Bluetooth packets. Plug-in modules therefore extend the system.

For example, the user can write a plug-in module to detect if any unauthenticated connections to a particular service or channel occurred by examining the stream for the appropriate packets. The user can write a plug-in module to monitor the count of a particular packet type in a stream of traffic. By aggregating the functions of several modules that have similar characteristics, the system increases the efficiency of the IDS engine.

5.4 Visualization Interface

The visualization interface of the system provides the administrator with the ability to observe events, traffic, and alerts in a convenient way. Visualization helps an administrator to distinguish between malicious activity and benign traffic. Within the field of intrusion detection, previous works have shown the benefit of utilizing visualization to detect attacks. [40, 41] Since the system detects only known attacks, visualization is useful in spotting anomalous, potentially malicious behavior. The implemented graphical interface provides details about the deviation between protocol layers, the specific operation codes for the LMP and L2CAP layers, and the activity on the RFCOMM and PSM channels. Figure 4 provides an example from the graphical user interface (GUI) of the implemented system.

Figure 4 depicts the traffic deviation between protocol layers during an RFCOMM scan attack on a Bluetooth target. The repeated pattern of traffic may give an administrator cause for inspection, by looking at the specific distribution of signaling codes on the L2CAP layer. By examining the specific L2CAP signaling codes, the administrator can detect an attacker attempting to repeatedly connect to RFCOMM channels.

Figure 4 additionally shows the RFCOMM channel activity of Bluetooth traffic captured during that period. In the previous 60 seconds, the attacker made 9 RFCOMM connection attempts, which is evidence of an active RFCOMM scan of a potential Bluetooth target.

5.5 Signature Development

Intrusion detection signatures for exploits can range from simple means, such as checking the header value of a field, to a highly complex stateful inspection or protocol analysis. Many Bluetooth attacks or exploits include purposely modified headers that violate the Bluetooth Core Specifications. Bluetooth protocol stack implementers have often assumed that the specifications would not be violated; hence the stacks are vulnerable to attack. The IDS signature

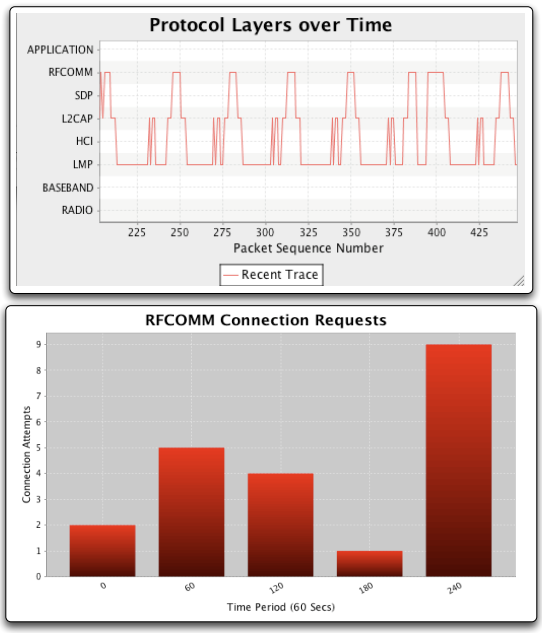


Figure 4. Visualization of RFCOMM Scan Attack and Bluetooth Channel Activity

can look at specific header fields or combinations of values to detect such attacks.

For more complex stateful inspections, the signatures use characteristics of an attack that are not easily evaded by an attacker. Unusual (legitimate, but suspicious) packets are best used in combination with other values to detect an attack. For an example, if a particular pattern of legitimate Bluetooth traffic causes a known failure on a particular device, then the IDS signature must take into account the device type of the intended target. Since Bluetooth devices respond to requests for their supported feature set, the IDS has this information at its disposal. An example is the HeloMoto attack, which plants a particular file on affected Motorola devices. By verifying that the intended target is a phone with certain feature characteristics, the signature can reduce but not necessarily eliminate false positives.

5.6 Response System

Once the system detects an attack signature, it has the capability to respond. This section examines different responses for each attack classification, including reconnaissance, denial of service, and information theft attacks. Directing responses requires careful consideration to avoid potential reflection attacks. This work does not address the security of responses but rather suggests methods to prevent, disrupt, and deny detected attacks.

5.6.1 Reconnaissance Response

For responding to a reconnaissance attack, this paper presents a method of deploying decoys or honeypots. By standing up honeypot targets, the system distracts attackers from more valuable machines on the network.[42] For a wireless IDS, the system should employ honeypots randomly in different locations so as not to give an attacker obvious knowledge of the boundaries of the wireless intrusion detection system. This work implements the response system on a separate machine; a deployed system would use multiple separate response nodes.

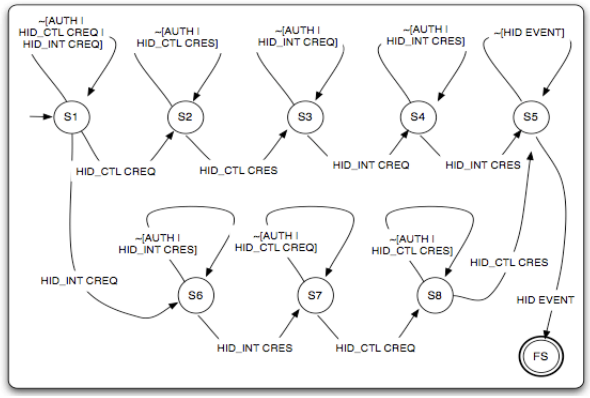
Upon detection of reconnaissance probes, the system responds by creating an array of fake devices to overwhelm and distract the attacker. The system creates false targets by randomly generating a name and physical address and flashing the information on a USB Bluetooth dongle. Once the device contains the new address, it responds to inquiries and name requests before burning a new randomly generated name and physical address onto the chipset. By utilizing a limited set of USB dongles, a low-cost system can create a mirage of many active Bluetooth devices to distract the attacker from legitimate Bluetooth devices.

5.6.2 Denial of Service Response

Denial of service attacks are responded to by terminating the connection between the attacker and target. A. Wool originally proposed the idea of using a false message to terminate legitimate Bluetooth traffic.[15] However, this work uses a similarly crafted message to terminate an attacker's traffic. Upon discovery of a denial of service attack, the IDS forges a message to disrupt communication between the attacker and the target. By sending the attacker an L2CAP CMD_REJ message with a forged address of the target, the IDS disrupts ongoing denial of service by the attacker.

5.6.3 Information Theft Response

Lastly, the system can respond actively to an information theft attack. Upon detection of such an attack, the response node stands up a false target device with the same physical address as the vulnerable device. Doing so provides a phony target device with services similar to the vulnerable device. Thus, the IDS slows down the the attacker who is attempting to connect to vulnerable targets, providing time for attacker identification and defensive measures.




```
[((~[AUTH | HID_CTL CREQ | HID_INT CREQ])*(HID_CTL CREQ))
(~[AUTH | HID_CTL CRES])*(HID_CTL CRES))
(~[AUTH | HID_INT CREQ])*(HID_INT CREQ))
(~[AUTH | HID_INT CRES])*(HID_INT CRES))) |
((~[AUTH | HID_CTL CREQ | HID_INT CREQ])*(HID_INT CREQ))
(~[AUTH | HID_INT CRES])*(HID_INT CRES))
(~[AUTH | HID_CTL CREQ])*(HID_CTL CREQ))
(~[AUTH | HID_CTL CRES])*(HID_CTL CRES)))]
[(HID_EVENT)]
```

Figure 5. Discrete Finite Automata and Corresponding Regular Expression for HIDAttack

5.7 Deterministic Finite Automata and Regular Expressions

Bluetooth attacks that require a stateful inspection can be modeled using a deterministic finite automata (DFA). This is illustrated in Figure 5. In this example, the finite state machine models an HIDattack that begins by accepting unauthenticated connections to the HID Control and Interrupt Channels, followed by a series of HID events on the Bluetooth protocol.

Current Intrusion Detection Systems such as Snort allow specifying attacks as regular expressions [29]. The IDS user must convert the DFA into an equivalent regular expression. Figure 5 depicts the same HIDAttack modeled as a regular expression. The authors are currently expanding this work to allow the implemented IDS to accept more complex regular expressions to detect attacks.

6 Implementation

6.1 Overview

This section explains the implementation and testing of the Bluetooth intrusion detection system. First, the testbed is described. Following that is a discussion of the practical problems and limitations faced when constructing the first network-based Bluetooth intrusion detection system.

6.2 Bluetooth IDS Testbed

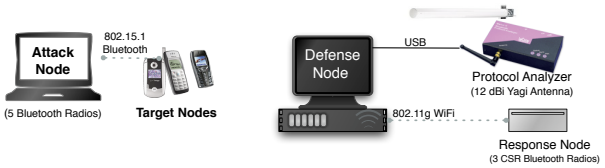


Figure 6. Testbed Used for Bluetooth Intrusion Detection

The testbed for this work consisted of a defense node, a response node, an attack node, and vulnerable target nodes. Figure 6 shows the testbed. The defense node was responsible for recording traffic, and identifying attacks initiated by the attack node on the target nodes. The response node attempted to disrupt, deny, and prevent Bluetooth attacks, as directed by the defense node.

6.2.1 Attack Node

The attack node consisted of a notebook computer running the BackTrack2 live Linux distribution from remote-exploit.org. Built on the Linux 2.6.20 kernel, the distribution includes more than 300 different security tools. Hackers employ BackTrack2 in order to penetrate computer security. The latest release includes existing support for 11 unique Bluetooth attacks. In addition, the testbed software included the Bluediving (next-generation Bluetooth security tool) available from bluediving.sourceforge.net. The Bluediving tool includes applications capable of spoofing Bluetooth addresses, generating L2CAP packets, and launching several of the attacks outlined previously. Furthermore, the attack software included the tools available from trifinite.org, an organization that hosts a large repository of Bluetooth attack tools. To augment the existing tools, the authors wrote several small programs to test the timing and frequency of differing reconnaissance probes for discoverable and non-discoverable devices. In addition to the Bluetooth radio included with the notebook computer, the attack node included five Bluetooth USB dongles and a modified Linksys USB110 Bluetooth dongle running in parallel to increase the probability of successful attacks. The attacks targeted all layers of the Bluetooth protocol stack.

Table 1. Targeted Devices

Attack	Target Description
BlueBug Attack	Nokia 6310 Phone
BlueSnarf Attack	Sony Ericsson T68i Phone
CarWhisperer Attack	Plantronics M2500 Headset
HeloMoto Attack	Motorola v600 Phone

6.2.2 Target Nodes

The targets in the experiment included a variety of devices with well documented design vulnerabilities. The experiment conducted reconnaissance and denial of service on a wide array of devices in order to establish a baseline of metrics to evaluate the implemented system. Furthermore, the experiment implemented specific attacks on devices with disclosed vulnerabilities. Table 1 outlines some attacks and the respective target devices.

6.2.3 Defense Node

The defense node consisted of a hardware protocol analyzer and a software IDS application. The Merlin LeCroy Protocol Analyzer nonintrusively captures, displays, and analyzes Bluetooth piconet data. The Merlin Analyzer also supports addressing the device via a scripting language. The scripting language enabled the system to listen to devices on a specific frequency or specific piconet. Further, it allowed recording and logging of traffic on all the Bluetooth protocol layers. Intended for Bluetooth developers, the analyzer included the ability to connect an external antenna. For purposes of this experiment, the system utilized a 12 dBI gain antenna to record traffic, with ranges up to 1 km.

The defense node also included a software application that processed the captured traffic, and ran a set of preconfigured rules and plug-in modules to detect Bluetooth-enabled attacks. The software application implemented the Bluetooth IDS engine. Additionally, the application included a graphical interface that provided the alert and visualization interfaces. Furthermore, the application provides the security administrator with the capability to configure rules and write add-on modules for more complex attack signatures.

6.2.4 Intrusion Response Node

To demonstrate response capabilities, an intrusion response node was designed to distract, deter and terminate Bluetooth attacks. The response node contained three Cambridge Silicon Radio (CSR) chip-based USB Bluetooth dongles. These devices contained flash memory that permitted raw access to the device. As such, the chipsets enabled writing false information to forge the identity of the Bluetooth MAC devices. Forging the Bluetooth address proved essential since it enabled the response node to deploy honeypots with false identities, disrupt ongoing attackers by spoofing an address of an attacker, and prevent future attacks by forging the connection responses of vulnerable targets.

6.3 Practical Problems Faced

Implementing the first network-based Bluetooth IDS included some practical problems. The system relied on a hardware protocol analyzer to capture and decode Bluetooth packets. Therefore, the system faced some problems decoding particular packets, and scheduling unique piconets.

6.3.1 Protocol Analyzer Packet Decoding

The hardware analyzer presented a problem in realtime testing of three specific attacks,because of the version of the analyzer. For the research, the authors utilized an older analyzer that could decode Bluetooth traffic compliant with the 1.1 Core Specification, but not for the most recent Bluetooth Core Specification. Specifically, the analyzer required specific channel decoding assignments to understand which protocol layer had generated RFCOMM traffic for some packets during the BlueSnarfer, BlueBugger, and HeloMoto attacks. Thus, the authors could only record these attacks manually, because they required additional input to the analyzer. The authors tested the remainder of the attacks autonomously utilizing the scripting language of the analyzers.

6.3.2 Scheduling Between Bluetooth Piconets

Additionally, the protocol analyzer presented a problem in the fact that it could not simultaneously record all Bluetooth communication in a given area. Rather, it records communication in only one unique piconet at a time. In theory, a Bluetooth IDS should simultaneously listen to 79 different frequencies and then assign each captured packet to a particular piconet. Such a system would require 79 unique radios and a multiplexing scheme. Alternatively, a user could attempt to synchronize with a unique piconet and hop in sequence with that particular piconet. Alternating between piconets provides an efficient, although less than optimal, picture of traffic. The possibility does exist that one could miss an attack during the alternation between piconets. The testbed for this work employed a system that scheduled unique piconets for recording in a round-robin algorithm.

7 Experimental Evaluation

7.1 Overview

This section evaluates the implemented intrusion detection system. The metrics for intrusion detection and response are those recommended by the Defense Advanced Research Projects Agency (DARPA).

7.2 Experiment Setup

The testbed described in the previous section was used for the experiments. To test the probability of detection, the authors generated traffic as suggested by Mell et. al. [43] Using 20 different attack tools, the attack node ran exploits against the series of vulnerable target nodes. Each attack was recorded by the Bluetooth Protocol Analyzer. Furthermore, A special analyzer recording options file reduced the amount of traffic uploaded to the IDS. To increase the efficiency of the system, the analyzer dropped all baseband and radio traffic. Additionally, decoding assignments for RFCOMM AT traffic were specified that the protocol analyzer could not automatically decode (due to the limitations mentioned above). The IDS then analyzed the off-line packet recordings using a complete set of signatures.

Following that, the attack node launched a set of 17 different attacks on the target nodes. Decoded packets were streamed in realtime from the protocol analyzer to the IDS preprocessor. During the on-line detection tests, the analyzer was instructed via the scripting language to upload decoded packets to the IDS. During this experiment, the analyzer only recorded data originating from the attack node, instead of round robin scheduling.

To verify the probability of false alarms, the authors used previously recorded benign traffic provided by the LeCroy Corporation. This traffic was generated by 33 different devices, including Bluetooth-enabled computers, headsets, phones, HID devices, and handheld computers. The benign traffic contained over 26,000 previously recorded packets. The IDS examined this collection of packets in less than 30 seconds.

To evaluate the response capability, the authors built a response node based on Linux and the BlueZ protocol stack. The IDS then contacted the response node via TCP sockets and issued commands based on attack identification. The results were then verified by examining the output and data recorded on the response node.

7.3 Evaluation of IDS Metrics

In 2007, DARPA defined metrics for intrusion detection systems, including recording the Probability of False Alarms, Probability of Detection, Resistance to Attacks Directed at the IDS, Ability to Detect Never Before Seen Attacks, Ability to Identify an Attack, and Ability to Determine Attack Success.[43]

We used these metrics to determine the success of the implemented intrusion detection system. The results show the system has a low rate of false alarms, a high rate of detection, a moderate resistance to IDS attacks, and the ability to determine attack success.

7.3.1 Coverage

Coverage defines the attacks that an intrusion detection system can detect under ideal conditions.[43] For signature-based systems, coverage defines the set of known, defined signatures. The coverage of the implemented system consists of 20 known attacks. However, because the system has a configurable rule syntax and the user can add additional modules, the coverage can grow as the number of discovered Bluetooth attacks grows.

7.3.2 Response Time

Table 2. Time Required for Attack Detection

Attack	Category	Avg. Required TWin (sec)
RFCOMM Scan	Reconnaissance	110.86
PSM Scan	Reconnaissance	4.747
HeaderOverFlow	Denial of Service	0.0006
Nasty vCard	Denial of Service	1.1030
BlueSnarf	Information Theft	1.4696
BlueBugger	Information Theft	3.2566
CarWhisperer	Information Theft	0.2277
HeloMoto	Information Theft	3.2294

Figure 2 shows the average time required (Twin) to detect different Bluetooth attacks recorded by the system. Some attacks, such as the HeaderOverFlow attack, occur very briefly in 625 milliseconds. Other attacks that require an inspection of multiple packets take a longer time to report. Thus, some attacks such as an RFCOMM Scan occur over a period of more than a minute. Based on these values, the authors selected a sliding window value of 120 seconds of traffic to ensure that the IDS could always spot the patterns for the known attacks. The system detects most attacks within a matter of seconds. Near realtime detection allows the system to direct a quick response to prevent, disrupt, or deny ongoing attacks by a hacker.

7.3.3 Probability of Detection

The probability of detection measures the rate of attacks detected correctly by an IDS in a given environment during a particular time frame.[43] To test the probability of detection, the authors attacked the target nodes with 20 different attacks. The IDS used the previously calculated sliding window of 120 seconds of traffic. Furthermore, the authors tested the system with both off-line and on-line detection, with the exception of three attacks that could only be tested offline. The IDS correctly identified each of the 20 attacks in the tests.

7.3.4 Probability of False Alarms

False alarms incorrectly produce alerts on benign background traffic.[43] To test the results of the system implemented here, the benign traffic described above was used. The IDS processed these recordings of packets offline to determine a rate of false positives. The system did not produce any false-negative alerts on any of the benign traffic.

7.3.5 Resistance to Attacks Directed at the IDS

Common intrusion detection evasion techniques include sending fragmented packets, crafting obfuscated payloads, and overwhelming the IDS with alerts to disguise the actual attack.[43] All of these attacks attempt to overwhelm the IDS with data in order to decrease its ability to make an intelligent decision. Fortunately, the limited bandwidth of Bluetooth decreases the likelihood and severity of such attacks. The use of a protocol analyzer prevents fragmented packets and obfuscated payloads from overwhelming the IDS, as the protocol analyzer handles the decoding of Bluetooth packets. Fragmented packets are reassembled by the protocol analyzer in a way that is transparent to the IDS engine.

However, a separate problem exists if an attacker has access to a large number of Bluetooth radios. The current proposed system detects devices and listens in a round-robin fashion to each device for a specified period. Knowing this, an attacker could use several devices to conceal his actual attack. However, this attack would have a limited range, as an attacker would not likely have access to several long-distance antennae. To avoid this problem, a system would need to simultaneously record and multiplex all 79 frequencies used by the Bluetooth protocol. This is strictly an issue of cost, not technical feasibility.

7.3.6 Ability to Identify an Attack

Another metric is the ability to correctly identify an attack with a specific common or vulnerability name. [43] In the implemented IDS, the system correctly identifies all defined attacks by name. Further, it can distinguish between different tools implementing the attacks. For example, the system correctly differentiates between BTScanner and Tbear device reconnaissance tools. Both tools scan for Bluetooth devices by continuously generating inquiry requests. However, the tools utilize a different inquiry response timeout value. The IDS distinguished between these two tools performing a similar attack on the basis of the inquiry timeout value.

7.3.7 Ability to Determine Attack Success

The proposed system correctly identifies the success of an attack. This section presents two different examples to illustrate the attack success determination.

In the first example, the attack nodes crashed a Nokia phone by sending a malicious packet. After the receipt of the malicious packet, the Bluetooth radio of the Nokia phone ceased working and stopped sending acknowledgments for connection-oriented packets. Seeing that the Bluetooth radio in the phone had ceased working, the IDS identified the attack as successful. The authors repeated the same experiment with a Motorola phone not vulnerable to the attack. As expected, the Motorola phone did not crash and continued to respond after the attack. The intrusion detection system recorded the packets of the non-vulnerable device and identified the attack as being unsuccessful.

Figure 7. Report of Data Stolen During BlueBug Attack

In the second example, the attack node attempted to pull data out of a phone by using a BlueBug attack. The data consisted of a phonebook with the names of the top ten wanted terrorists. First, the attack node connected to a vulnerable phone and issued the command to steal the phonebook. The intrusion detection system correctly saw the packets relevant to the connection and the data packets of the stolen phonebook. Thus, it identified the attack as successful. Figure 7 shows the GUI alert. The authors then repeated the same experiment with a non-vulnerable phone. The intrusion detection system saw the connection and attempt to pull the phonebook. However, the attack failed to

steal the phonebook, and the intrusion detection successfully reported an unsuccessful attack, since it did not see any data for the phonebook.

7.3.8 Processing Speed

Table 3. Off-line Processing Time

Attack	Traffic Length (sec)	Packets	Processing Time (ms)
BlueBugger	30.02	189	21.0
BSS	367.20	973	144.0
BlueSpam	358.88	1,756	165.0
PSM Scan	47.85	2,187	292.0

One measurement for determining the relative responsiveness of an intrusion detection system is the speed at which the IDS can process traffic. Table 3 shows traffic processed off-line by the IDS, and the time required to match the entire set of rules against the different traffic sets. The results show that the system can process roughly 1,000 packets of data in roughly one second. It is important to note that the term *packet* here describes logical packets, not physical packets, since the protocol analyzer combines several packet fragments together and exports them as one logical packet. Further, the analyzer does not export any baseband or radio layer traffic, since the IDS system is only concerned with identifying the malicious activity on the upper layers of the Bluetooth protocol.

7.4 Evaluation of Response Capability

The results of responding to attacks were also tested and measured. Through the use of flash-enabled Bluetooth radios, the system created honeypots to distract attackers and sent falsified messages to terminate attacks.

7.4.1 Reconnaissance Response

Honeypots have been used effectively to distract attackers from IP targets.[42] However, this section describes the results of using honeypots to distract mobile attackers on the Bluetooth protocol. An advantage of Bluetooth attacks over typical wireless attacks is the relatively quick time in which an attacker can find and comprise a target. These results present the success of one method of aggressively responding to a reconnaissance threat by flooding the attacker with false honeypot targets in order to increase the time required for an attacker to find a target device.

To confuse an attacker, the IDS used the following technique. Upon notification of a reconnaissance probe, the system responded by deploying three Bluetooth radios that constantly changed their user-friendly names and physical MAC addresses. Thus, it appeared to the attacker that there was a large volume of Bluetooth targets. In order to create the mirage of targets, the system flashed the chipsets of each Bluetooth radio. On average, it took 5.7947 seconds to flash the chipset, reboot the chip and change the device name after responding to an inquiry. Thus, the system essentially created ten false targets per radio per minute.

To verify the results, the authors wrote a reconnaissance program that recorded the names of unique targets detected per minute. On average, the inquiry program detected only 4 false targets per additional radio per minute. This lower value corresponds to an inquiry period of 10.24 seconds plus the short period during which an attacker must ask for the user-friendly name of the Bluetooth device. At a cost of $3 per Bluetooth radio, a production system could easily employ hundreds of Bluetooth radios to quickly distract an attacker. Further, the system could place these throughout an organization to distract attackers.

7.4.2 Denial of Service Responses

The authors further tested the IDS response system by attempting to break ongoing denial of service attacks using falsified connection termination messages.[15] In order to utilize this attack method, the attack must know both the physical addresses of the attacker (master) and target device (slave) in the piconet. The system already had knowledge of the master address from discovery by the protocol analyzer. Had the system been implemented using another method, the master address would also have been available as a field in FHS packet sent at the start of the connection. However, gaining access to the target address proved more challenging. The packet header only includes a 3-bit AMA address referring to the slave's position in the piconet. Thus, the response node had to scan for all discoverable devices. The response system then forged packets from all of those devices to the attacker.

The system then attempted to stop an attacker attacking via the BlueSmack Attack. The system successfully terminated a denial of service attack. At the attacker's console, the denial of service program reported a connection timeout and disconnected from the target, stopping the attack. Further testing showed that the response capability disrupted similar attacks such as Tanya, Ping of Death, and Symbian Remote Restart attacks.

7.4.3 Information Theft Responses

Next, the authors tested the ability of the system to respond to information theft attacks such as the BlueSnarf, BlueBug, CarWhisperer, and HeloMoto attacks, by establishing false targets. Establishing false targets protects the vulnerable

targets by creating phony devices with the same physical address as the vulnerable target.

To test the defense, the authors simply flashed a radio with the same physical address as a vulnerable device. The authors then attempted to perform an attack against the vulnerable device. Because the flashed device had a higher power class, it answered and generated replies for the traffic intended for the vulnerable target. This caused the attack to fail on all four attack methods above. The more powerful phony-target-response device responded to the message instead of the intended target. Based on the success of this test, the implemented system could be further expanded by creating phony services that supplied false information to attackers. However, this method of response is limited in the fact that it unfortunately causes a denial of service attack in the process.

8 Conclusion

Bluetooth is becoming a ubiquitous protocol. While standard applications include smartphones, hands-free audio, global positioning devices, cameras, and peripheral cable replacement, Bluetooth devices also exist in health care, mobile banking, and military applications. Bluetooth-enabled devices now carry sensitive information, attracting hackers.

The lack of mandatory authentication, a weak encryption key scheme, and differing vendor protocol implementations have created the possibility for several attacks on Bluetooth devices. Recent trends have shown an increase in attacks on Bluetooth-enabled devices and the combination of other attacks implemented over Bluetooth. Furthermore, the ease of implementing Bluetooth attacks has decreased with the proliferation of several tools and online repositories of information.

This paper presents a network intrusion detection system, based on misuse detection, to detect Bluetooth attacks. This system has the limitation of all misuse detection schemes, which is the inability to automatically new categories of attacks. However, the system provides an efficient and effective means of detecting known intrusions. Furthermore, the system demonstrates the ability to determine the success of an attack, and is resistant to attacks on the IDS itself.

This work demonstrates that a Bluetooth intrusion detection system can actively respond to threats. This work presents a means to distract attackers via the use of a mirage of Bluetooth devices. Further, this work implements a system for stopping ongoing attacks through specially crafted messages.

Limitations of the work and possible future directions include:

1. Inability to automatically recognize new classes of attack. The potential of anomaly detection for Bluetooth remains to be investigated.

2. Fusing data from IDS sensors on different piconets might provide a broader view of attack strategies.

3. Identifying the geographic location of the attacker might be possible using the technique of Rodriguez et al.[44]. This would provide a wider scope for effective intrusion responses.

4. Integration of the IDS with other Bluetooth security assessment toolsuites.

References

[1] "Installed base of more than one billion products gives consumers more than five billion ways to use the global wireless standard," The Bluetooth Special Interest Group (SIG), Tech. Rep., Nov 2006.

[2] F. Thomson, "Bluetooth enabled equipment shipments to hit 800 million this year," IMS Research, Tech. Rep., Oct 2007.

[3] B. SIG, "Core specification v2.0 + EDR," Bluetooth SIG, Tech. Rep., Nov 2004.

[4] D. Cypher, N. Chevrollier, N. Montavont, and N. Golmie, "Prevailing over wires in healthcare environments: Benefits and challenges," *Communications Magazine, IEEE*, vol. 44, no. 4, pp. 56–63, Apr 2006.

[5] D. Beaumont, "Bluetooth brings mobility to health care," *Planet Wireless*, pp. 11–15, 2002.

[6] D. Miller, "Mobile finance: Pay as you go," *Unwired Magazine*, vol. 5, pp. 16–17, 2007.

[7] "Mexican bank deploys hypercom bluetooth-enabled payment stations," *Mobile Enterprise Magazine*, Oct 2007.

[8] B. Brewin, "AirDefense sniffs out Bank of America Bluetooth-based ID system," *Computer World*, May 2004.

[9] K. Kaye, "Navy campaign takes Bluetooth plunge," *ClickZ News*, 2007.

[10] "DoD wireless push email system security requirements matrix," DISA Field Security Operations (FSO), Tech. Rep. 2.0, Jun 2007.

[11] "BAA 07-46 Proposer Information Pamphet (PIP): Landroids," Defense Advanced Research Projects Agency (DARPA), Tech. Rep., Jul 2007.

[12] O. Holland, J. Woods, R. DeNardi, and A. Clark, "Beyond swarm intelligence: the ultraswarm," in *IEEE Swarm Intelligence Symposium SIS2005*, Jun 2005, pp. 217–224.

[13] H. Everett, E. Pacis, G. Kogut, N. Farrington, and S. Khurana, "Toward a warfighter's associate: Eliminating the operator control unit," in *SPIE Proceedings 5609: Mobile Robots XVII*, Oct 2004.

[14] "Core specification v2.1 + EDR," Bluetooth Special Interests Group (SIG), Tech. Rep., Aug 2007.

[15] Y. Shaked and A. Wool, "Cracking the bluetooth pin," in *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM, Jun 2005, pp. 39–50.

[16] O. Levy and A. Wool, "A uniform framework for cryptanalysis of the bluetooth e0 cipher," in *1st International Conference on Security and Privacy for Emerging Areas in Communication networks (SecureComm'05)*, Sep 2005, pp. 365–373.

[17] "Simple pairing whitepaper," Bluetooth Special Interests Group (SIG), Core Specification Working Group, Tech. Rep. Release Version V10r00, Aug 2006.

[18] T. Karygiannis, "Wireless network security: 802.11, bluetooth and handheld devices," NIST, Tech. Rep., Nov 2002.

[19] C. Mulliner, "HID attack (attacking HID host implementation)," http://www.mulliner.org/bluetooth/hidattack.php.

[20] M. Herfurt, "Carwhisperer," http://trifinite.org/trifinite_stuff_carwhisperer.html.

[21] ——, "Bluesnarfing @ CeBIT 2004: Detecting and attacking bluetooth-enabled cellphones at the Hannover Fairground," in *Salzburg Research Forschungsgesellschaft*, Mar 2004, pp. 1–12.

[22] K. Mahaffey and J. Hering, http://www.flexilis.com/.

[23] T. Buennemeyer, T. Nelson, M. Gora, R. Marchany, and J. Trong, "Battery polling and trace determination for bluetooth attack detection in mobile devices," in *Information Assurance and Security Workshop, 2007. IAW '07. IEEE SMC*, Jun 2007, pp. 135–142.

[24] Y. Lim, T. Yer, J. Levine, and H. Owen, "Wireless intrusion detection and response," in *Information Assurance Workshop, 2003. IEEE Systems, Man and Cybernetics Society*. IEEE, Jun 2003, pp. 68–75.

[25] J. Anderson, "Computer security threat monitoring and surveillance," James P. Anderson Co., Tech. Rep., 1980.

[26] D. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. 13, no. 2, pp. 222–232, February 1987.

[27] S. Kumar and E. H. Spafford, "A Pattern Matching Model for Misuse Intrusion Detection," in *Proceedings of the 17th National Computer Security Conference*, 1994, pp. 11–21.

[28] A. K. Ghosh and A. Schwartzbard, "A study in using neural networks for anomaly and misuse detection," in *SSYM'99: Proceedings of the 8th conference on USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 1999, pp. 12–12.

[29] M. Roesch, "SNORT - lightweight intrusion detection for networks," in *13th LISA Conference*, Nov 1999, pp. 229–238.

[30] "Fortress dominates wireless security market protecting over 10,000 networks," http://www.fortresstech.com, December 2004.

[31] D. Welch, "Wireless security threat taxonomy," in *2003 IEEE Workshop on Information Assurance*, United States Military Academy, West Point, NY, Jun 2003.

[32] E. Bierman and E. Cloete, "Classification of malicious host threats in mobile agent computing," in *Proceedings of the 2002 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology*, vol. 30, 2002, pp. 141–148.

[33] J. Cache and V. Liu, *Hacking Wireless Exposed*. McGraw-Hill, 2007.

[34] A. Laurie, "Helomoto attack," http://trifinite.org/trifinite_stuff_helomoto.html.

[35] K. Haataja, "Bluetooth network vulnerability to disclosure, integrity and denial-of-service attacks," in *Proceedings of the Annual Finnish Data Processing Week at the University of Petrozavodsk (FDPW'2005)*, vol. 7, 2005, pp. 63–103.

[36] G. Yan, L. Cuellar, S. Eidenbenz, H. D. Flores, N. Hengartner, and V. Vu, "Bluetooth worm propagation: Mobility pattern matters!" in *2nd ACM symposium on information, computer and communications security*. ACM, 2007, pp. 32–44.

[37] D. Spill and A. Bittau, "Bluesniff: eve meets alice and Bluetooth," in *WOOT'07: Proceedings of the first conference on First USENIX Workshop on Offensive Technologies*. USENIX Association, Aug 2007.

[38] A. Orebaugh, G. Ramirez, J. Burke, and L. Pesce, *Wireshark & Ethereal Network Protocol Analyzer Toolkit (Jay Beale's Open Source Security)*. Syngress Publishing, 2006.

[39] "Enterprise wireless monitoring for bluetooth networks (AirDefense BlueWatch)," Air Defense Inc, Tech. Rep., 2005.

[40] A. Oline and D. Reiners, "Exploring three-dimensional visualization for intrusion detection," in *Visualization for Computer Security, 2005. (VizSEC 05).*, Oct 2005, pp. 113–120.

[41] K. Abdullah, C. Lee, G. Conti, and J. Copeland, "Visualizing network data for intrusion detection," in *Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC*, Jun 2005, pp. 100–108.

[42] I. Mokube and M. Adams, "Honeypots: concepts, approaches, and challenges," in *Proceedings of the 45th annual southeast regional conference*, SIGAPP: ACM Special Interest Group on Applied Computing. ACM, 2007, pp. 321–326.

[43] P. Mell, V. Hu, R. Lipmann, J. Haines, and M. Zissman, "An overview of issues in testing intrusion detection systems," National Institute of Standards and Technology, Tech. Rep., 2007.

[44] M. Rodriguez, J. Pece, and C. J. Escudero, "In-building location using bluetooth," in *International Workshop on Wireless Ad-hoc Networks (IWANN'05)*, May 2005.