

ABSTRACT

OCONNOR, TERRENCE J. Bluetooth Intrusion Detection. (Under the direction of Professor Douglas Reeves),

Bluetooth, a protocol designed to replace peripheral cables, has grown steadily over the last five years and includes a variety of applications. In near ubiquity now, the Bluetooth protocol operates on a wide variety of mobile and wireless devices. Several attacks exist that successfully target and exploit Bluetooth enabled devices. This thesis describes the implementation of a network intrusion detection system for discovering malicious Bluetooth traffic.

This work improves upon existing techniques, which only detect only a limited set of known attacks through measuring anomalies in the power levels of the Bluetooth device. This research illustrates the ability to efficiently and effectively detect a wide variety of malicious traffic by utilizing pattern matching misuse detection. This system identifies reconnaissance, denial of service, and information theft attacks on Bluetooth enabled devices. In addition, this tool includes a visualization interface to facilitate the understanding of Bluetooth enabled attacks. Furthermore, this system implements an intrusion response based on attack classification.

This thesis presents the implementation of the Bluetooth Intrusion Detection System and demonstrates its detection, analysis, and response capabilities. The experimental results show that the system can significantly improve the overall security of an organization by identifying and responding to threats posed on the Bluetooth protocol.

Bluetooth Intrusion Detection

by

Terrence J. OConnor

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh, North Carolina

2008

Approved By:

Dr. Vincent W. Freeh

Dr. Peng Ning

Dr. Douglas Reeves
Chair of Advisory Committee

DEDICATION

To CR and PM who gave their lives so that others may live. You are my heroes.
Your sacrifices are not forgotten.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dr. Douglas Reeves, Dr. Peng Ning, and Dr. Vincent Freeh, who provided a constant source of guidance, motivation, and support. Furthermore, I would like to thank Dr. Douglas Reeves for his mentorship in teaching. Additionally, I'd like to thank Nan Schweiger for proofreading this document. Lastly, I'd like to thank my family for their support during this process.

TABLE OF CONTENTS

LIST OF TABLES	vii
LIST OF FIGURES	viii
1 Introduction	1
1.1 Bluetooth-Enabled Technology	1
1.2 Motivation	1
1.2.1 Health Care Bluetooth-Enabled Technology	2
1.2.2 Financial Sector Bluetooth-Enabled Technology	2
1.2.3 Military Bluetooth-Enabled Technology	3
1.3 Summary of Contribution	4
1.4 Thesis Organization	4
2 Background	5
2.1 Bluetooth-Enabled Attacks	5
2.1.1 Long Distance Attacks	5
2.1.2 Reconnaissance	6
2.1.3 Denial of Service	9
2.1.4 Information Theft	11
2.1.5 Emerging Trends	14
2.2 Intrusion Detection	15
2.2.1 Network Intrusion Detection Systems	15
2.2.2 Anomaly-Based Intrusion Detection	16
2.2.3 Misuse Intrusion Detection	18
2.2.4 Realtime Intrusion Detection	19
2.2.5 Visualization of Attacks	20
2.2.6 Active Response	21
3 Related Work	22
3.1 Intrusion Detection	22
3.2 Wireless and Mobile Threat Modeling	23
3.3 Current Related Tools	24
4 Bluetooth IDS Design	25
4.1 Overview	25
4.2 Packet Decoding and Preprocessing	26
4.3 IDS Engine	27

4.3.1	Pattern Matching	27
4.3.2	Plug-in Modules	28
4.3.3	IDS Engine Rule Grammar	28
4.4	Visualization Interface	28
4.5	Signature Development	31
4.6	Signatures for Reconnaissance Attacks	32
4.6.1	Signature of Device Discovery Attack	32
4.6.2	Signature of Service Discovery Attack	33
4.6.3	Signature of Exploit Discovery Attack	34
4.7	Detection of Denial of Service Attacks	35
4.7.1	Signature of BlueSpamming	35
4.7.2	Signature of Tanya Attack	36
4.7.3	Signatures of Maliciously Formed Bluetooth Packets	37
4.8	Detection of Information Theft Attacks	38
4.8.1	Signature of Btcrack	38
4.8.2	Signature of CarWhisperer Attack	39
4.8.3	Signature of BlueBug Attack	40
4.8.4	Signature of HeloMoto attack	41
4.8.5	Signature of HIDattack	41
4.9	Response System	42
4.9.1	Reconnaissance Response	42
4.9.2	Denial of Service Response	43
4.9.3	Information Theft Response	43
5	Bluetooth IDS Implementation	45
5.1	Overview	45
5.2	Bluetooth IDS Testbed	45
5.2.1	Attack Node	46
5.2.2	Target Nodes	48
5.2.3	Defense Node	48
5.2.4	Intrusion Response Node	50
5.3	Practical Problems Faced	50
5.3.1	Protocol Analyzer Packet Decoding	51
5.3.2	Scheduling Between Bluetooth Piconets	51
6	Evaluation of Results	52
6.1	Overview	52
6.2	Experiment Setup	52
6.3	Evaluation of IDS Metrics	53
6.3.1	Coverage	53
6.3.2	Response Time	54
6.3.3	Probability of Detection	54

6.3.4	Probability of False Alarms	55
6.3.5	Resistance to Attacks Directed at the IDS	55
6.3.6	Ability to Identify an Attack	56
6.3.7	Ability to Determine Attack Success	56
6.3.8	Processing Speed	57
6.4	Evaluation of Response Capability	58
6.4.1	Reconnaissance Response	58
6.4.2	Denial of Service Responses	59
6.4.3	Information Theft Responses	60
7	Conclusion	61
7.1	Summary	61
7.2	Limitations and Future Work	62
7.2.1	Hybrid Model for Detection	62
7.2.2	Correlation of Multiple Bluetooth Sensors	62
7.2.3	Further Intrusion Response Development	63
7.2.4	Integration with Existing Defense Tools	63
7.2.5	Integration with Existing Intrusion Detection Systems	64
7.3	Speculation	64
7.3.1	Attacks Against the 2.1 Protocol	64
7.3.2	Attacks Against Infrastructure	65
	Bibliography	66
	Appendix A Bluetooth Primer	74
A.1	Bluetooth	74
A.1.1	Overview	74
A.1.2	Protocol Stack	77
A.1.3	Application Profiles	79
A.1.4	Device Discovery	79
A.1.5	Pairing and Authentication Process	80
A.1.6	Security	83
	Appendix B User's Manual	85
B.1	Intrusion Detection Graphical User Interface	85
B.1.1	Rule Configuration	86
B.1.2	Response Actions	86
B.1.3	Statistics Analysis	87
B.2	Example of a Bluetooth Module	89
B.3	Script for Analyzer Control	89
	Appendix C Bluetooth Attack Information	90

LIST OF TABLES

Table 5.1 Table of Bluetooth-Enabled Attacks	47
Table 5.2 Targeted Devices	48
Table 6.1 Time Required for Detection of Attacks.....	54
Table 6.2 Off-line Processing Time.....	57
Table A.1Bluetooth Protocol Stack Vulnerabilities.....	79

LIST OF FIGURES

Figure 2.1 Modification of Bluetooth MAC Address under Linux BlueZ Stack ..	11
Figure 2.2 Traffic Deviations Between Attack (BlueSmack) and Baseline Traffic	17
Figure 4.1 Design of Bluetooth Intrusion Detection System	25
Figure 4.2 Packet Reassembly by the Preprocessor	26
Figure 4.3 Rule to Detect Malicious L2CAP Header Overflow Attack	27
Figure 4.4 Bluetooth Intrusion Detection Grammar	29
Figure 4.5 Visualization of RFCOMM Scan Attack	30
Figure 4.6 Visualization of Bluetooth Channel Activity	31
Figure 4.7 Model for Tbear Reconnaissance Attack	33
Figure 4.8 Model for a RFCOMM Scan Reconnaissance Attack	34
Figure 4.9 Bluetooth Stack Smasher Reconnaissance Attack.....	35
Figure 4.10 Capture of Nasty vCard Attack.....	36
Figure 4.11 L2CAP Packet used in Tanya Attack	36
Figure 4.12 Nokia N70 Denial of Service Attack.....	37
Figure 4.13 L2CAP Header Overflow Denial of Service Attack.....	38
Figure 4.14 Capture of a Packet Used to Reset and a Connection for BTCrack ..	38
Figure 4.15 CarWhisperer Packet Transmission	39
Figure 4.16 BlueBug Attack Traffic Captured by IDS	40
Figure 4.17 Pattern for HeloMoto Traffic	41
Figure 4.18 Pattern for HIDattack Traffic	42

Figure 5.1 Testbed Used for Bluetooth Intrusion Detection.....	45
Figure 5.2 Default Rule List.....	49
Figure 6.1 Report of Data Stolen During BlueBug Attack.....	57
Figure 6.2 Attacker’s Console after Denial of Service Response.....	59
Figure 6.3 Attacker’s Console after Information Theft Response.....	60
Figure A.1 Example of a Bluetooth Scatternet	75
Figure A.2 Example for a Bluetooth Baseband Packet.....	76
Figure A.3 The Bluetooth Protocol Stack.....	77
Figure A.4 Link Manager Protocol (LMP) Classes of Bluetooth Power.....	78
Figure A.5 Bluetooth MAC Address.....	80
Figure A.6 Pairing and Mutual Authentication Prior to Core Specification 2.1..	81
Figure A.7 Creation of Diffie Helman Key used for Secure Simple Pairing in Core Specification 2.1	82
Figure A.8 Link Manager Protocol Security Modes.....	83
Figure B.1 Graphical User Interface.....	85
Figure B.2 Graphical User Interface.....	86
Figure B.3 Graphical User Interface.....	87
Figure B.4 Graphical User Interface.....	87
Figure B.5 Example of a Bluetooth Module	88

Chapter 1

Introduction

1.1 Bluetooth-Enabled Technology

The Bluetooth Special Interest Group developed the Bluetooth, (IEEE 802.15.1 protocol), as a communications protocol 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. This thesis proposes a system for detecting malicious use of the Bluetooth communications protocol.

1.2 Motivation

Bluetooth-enabled devices extend to many critical applications in industries outside of telecommunications. Examples include the health care, banking, and military applications. This section reviews these applications to understand the threat posed by Bluetooth-enabled attacks to their respective critical infrastructures.

1.2.1 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 provide patient mobility. Bluetooth serves as an attractive protocol for health care administrators because of the 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.

1.2.2 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.

1.2.3 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.

1.3 Summary of Contribution

The main contribution of this thesis is to design and implement an intrusion detection system that provides:

- The first documented network intrusion detection system (IDS) for malicious Bluetooth traffic.
- A well-defined set of rules and modules for known attacks.
- Several interactive analysis tools to analyze captured attacks.
- A comprehensive recording base of several Bluetooth attacks.
- An intrusion response capability for malicious Bluetooth attacks.

1.4 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 classifies the different types of Bluetooth attacks, and reviews the function of intrusion detection systems. Chapter 3 proposes the design of a system for detecting Bluetooth threats. Chapter 4 introduces the methodology behind the testing of the proposed IDS. Finally, Chapter 5 records the results of the testing and makes observations before the conclusion.

Chapter 2

Background

This chapter provides a background on attacks on the Bluetooth enabled devices and Intrusion Detection Systems (IDS). The first section classifies the threats posed by Bluetooth-enabled attacks. The second section provides an overview of intrusion detection monitoring schemes, detection techniques, visualization schemes and active responses. For a further explanation of the Bluetooth protocol specifications, Appendix A provides a Bluetooth primer.

2.1 Bluetooth-Enabled Attacks

This section examines some existing attacks on Bluetooth-enabled devices. Further, this section classifies threats into three different categories, including device reconnaissance, denial of service, and information theft. By classifying the different attacks and tools into different categories, this work enables the system to direct appropriate responses.

2.1.1 Long Distance Attacks

Prior to examining the attacks, this chapter reviews the motivation behind some of the Bluetooth attacks. Although Bluetooth was initially developed for ranges up to 100 meters, commercial products now exist to extend the range of Bluetooth

to 30 km. By modifying the antenna, power, and sensitivity of the device radio, several researchers have shown successful Bluetooth connections over extended ranges. Herfurt et al. demonstrated device exploitation over a 1.7 km range as early as 2004.[14] Among Bluetooth hackers, the Linksys USBBT100 Bluetooth dongle serves as a popular device for modification.[15] An attacker can pry open the device and replace it with a more powerful high-gain antenna with specific capabilities and effects. Thus, a Bluetooth hacker can attack devices from a comfortable range away from the target. In building the attack testbed for this work, the author constructed a modified Linksys dongle and attached a 7 dBi gain whip antenna, 12 dBi gain Yagi directional antenna, and a 12 dBi gain omni-directional di-pole antenna. With a successful long-range Bluetooth device, a hacker may conduct reconnaissance of Bluetooth-enabled devices.

2.1.2 Reconnaissance

Device reconnaissance includes any type of attack that attempts to gather information about a Bluetooth-enabled device in order to proceed with further attacks. Successful reconnaissance detection allows targets to take countermeasures prior to follow-on attacks. For a Bluetooth attacker, the first step in reconnaissance consists of device discovery. This chapter examines three different methods of discovering devices. The methods include broadcasting inquiries, brute-force detection, and passive listening to a specific frequency for traffic. Once an attacker has discovered a device, he may proceed with further reconnaissance including location tracking, exploit discovery, or service discovery.

Device Discovery

In the first method of device discovery, a device continually broadcasts inquiry requests until receiving successful acknowledgments. An attacker can mount this attack through a single radio or increase the probability of success by utilizing multiple radios. In either case, each radio sends an inquiry request and then requires a minimum of 10.24 seconds to collect all inquiry responses.[16] Implementations of this attack

include btscanner, btcrawler, greenplaque, tbsearch, and ghattotooth.[17] To demonstrate the lethality of such a reconnaissance attack, Van Es of Bluetoothtracking.org established a node in Apeldoorn, Netherlands, that repeatedly sent inquiry requests. Between September and December of 2007, Van Es received replies from 45,953 unique Bluetooth-enabled devices.[18] In addition, F-Secure demonstrated the capability to detect 1,405 unique Bluetooth devices over a 7-day experiment.[19] While capable of finding several discoverable devices, inquiry requests provide no information about non-discoverable devices.

An attacker must use a separate method to find non-discoverable Bluetooth devices. Although non-discoverable devices do not respond to inquiry responses, they still respond to any remote name request messages. However, an attacker must know the Medium Access Control (MAC) address of a device to generate a unicast remote name request. The Bluetooth MAC address consists of 24 vendor-specific bits and 24 bits unique to the model and device. Online databases of MAC addresses for common devices exist.[20] With knowledge of the first 24 bits of the MAC, an attacker can prepare a limited brute-force attack to discover the remaining bits of the specific model and device. Such attacks will send repeated remote name requests to different MAC addresses until a valid response confirming the presence of a device is received. O. Whitehouse of Symantec Security developed an implementation for this attack that additionally utilizes multiple radios for increased probability of successful discovery.[21]

The final method introduces a means of passively discovering devices in contrast to the two previous aggressive methods. Spill and Bittau demonstrate a method of passively discovering devices, utilizing a software-defined radio (USRP) that listens for multiple frames on a single frequency.[20] Spill and Bittau dubbed this method BlueSniff since the radio essentially sniffs for Bluetooth frames. Although each frame contains only part of the MAC address, it contains an error correction code based on the entire MAC. Thus, they demonstrate a technique for discovering the remaining bits of the MAC. In this approach, the reconnaissance device generates no traffic and therefore operates in a passive and somewhat undetectable mode. However, to mount any future any attack, the attackers must additionally know the clock offset.

To discover the offset, Spill and Bittau acknowledge a Bluetooth attacker must make a connection with the device.[20] During that connection, the attacker can generate a read remote name request or read clock offset request.

Location Tracking via Bluetooth

The previous section demonstrated that Bluetooth-enabled devices suffer privacy issues because they will respond to certain requests regardless of the security or discoverability modes. This enables hackers to track movements of all Bluetooth-enabled devices by having knowledge of the device's MAC address.[17] A hacker can simply create a script that pings a known MAC address and reports when it receives positive acknowledgments. Further, the attacker can use this information to create patterns of travel. Van Es of Bluetoothtracking.org has successfully done this with 16 devices that he has watched travel over 85 miles.[18]

Alongside the location, this type of attack may reveal other information. In the worst example, Apple computers use the owner's name and computer type to determine the name of the Bluetooth device.[22] Generally, the vendor uses at least the product and model name. Thus, a hacker potentially knows who you are, the type of machine you are using, and the times you have been at a location. Working implementations of this attack include BlueAlert, BlueFang, and BlueFish.[17]

Service Discovery

With knowledge of a specific MAC address, an attacker may wish to discover the device type, channels, and services a device offers to carry out future attacks. The Protocol Service Multiplexer (PSM) multiplexes services for the Bluetooth Logical Link Control and Adaptation Protocol (L2CAP) layer. In the Bluetooth protocol, 65,535 PSM ports and 30 RFCOMM channels exist.[23] A service discovery or audit tool examines these channels and ports for services. Collin Mulliner of the Trifinite Group created the BT Audit tool to do such auditing.[23] Additionally, the Bloover does the same auditing but runs exclusively on handheld JSR-82-compliant devices. This attack is the IP equivalent of a portscan attack, looking for vulnerable, suspect,

or misconfigured services running on a device.

Exploit Discovery

Tools exist to examine Bluetooth-enabled devices' discoverable vulnerabilities and exploitability. Most notably, AirDefense markets a commercial Bluetooth scanner that identifies misconfigured and vulnerable Bluetooth devices in organizations.[24] The tool, dubbed BlueWatch, enables companies to take proactive steps to identify threats to an organization prior to attack.

Further, several open source efforts to perform L2CAP and RFCOMM automated, random testing of a wide array of devices exist. Security experts often refer to automated, random testing as fuzzing. The Bluetooth Stack Smasher (BSS) appears as one implementation of a Bluetooth fuzzer.[25] BSS discovers exploits by constructing specially crafted L2CAP messages.[25] Once the tool detects an instance of an exploit, it records the sequence of packets that caused the specific vulnerability. An online repository of known vulnerabilities discovered by BSS exists. With a known device or MAC address, an attacker can create a blueprint and model to attack a device.

2.1.3 Denial of Service

Bluetooth devices operate in the military, hospitals, and other very sensitive environments. Recent work shows Bluetooth communications in military androids, unmanned helicopters, robots, ruggedized military phones, and sensitive military communications.[11, 12, 10] Thus, a denial of service attack can have drastic consequences.

This thesis classifies Bluetooth denial of service attacks into four classes. The first class consists of BlueJacking. The second class involves maliciously crafted messages that attack particular vulnerabilities of devices. The third class consists of attacks that attempt to deplete the battery life of Bluetooth-enabled devices. The final class comprises Bluetooth worms that quickly spread, aided by the Bluetooth discovery methods and object exchange profile.

BlueJacking

BlueJacking consists of sending unsolicited messages via the Bluetooth protocol.[17] Because certain applications such as the Object Exchange Push Profile require no authentication, an attacker can simply distribute unrequested content to target devices. The distributed content can consist of text, images, video, or sound. Because certain image libraries have shown vulnerabilities, this attack can be used to distribute viruses or trojan horses.[26] Further, it can be coupled with long-range attacks to exploit a multitude of targets.[14]

Maliciously Crafted Messages

Maliciously crafted message attacks rely on specific vendor vulnerabilities. Similar to IP Pings of Death, some attacks send repeated L2CAP echo requests of a large size or over an extended period of time. Tanya and BlueSmack prove to be working implementations of this attack.[27] Further, Bluetooth attackers have reported that Nokia and Sony Ericsson devices prove susceptible to incorrectly assembled L2CAP packets. By modifying the L2CAP signal length field to an invalid size, an attacker can disrupt Bluetooth communications on these devices until the user performs a hard reset. Other malicious message attacks exploit particular vendor weaknesses by specially crafting vCard messages that force a device to fault.

Battery Depletion Attacks

Buennemeyer et al. originally proposed attacks to deplete the battery life of mobile devices by utilizing features of the Bluetooth protocol.[28] They created three attacks known as BlueSYN, BlueSYN Calling, and PingBlender that deplete the battery life of target devices. The attacks saturate the communication channel of the target device with unsolicited messages.[28] As a result, the attacks exhaust the Bluetooth resources of a target device. Further, the target device has no knowledge it is under attack.[28]

Bluetooth Worms

Inqtana provided the first proof of Bluetooth worms. The Inqtana worm specifically targets a vulnerability in the stack of the Apple OS X operating system. Once it infects a Bluetooth-enabled Apple computer, it attempts to find other devices and propagate. Because of the inherent device discovery tools of Bluetooth, the propagation of a worm appears a legitimate threat. Yan et al. proposed a model for examining Bluetooth worm propagation.[29] Additionally, the Cabir and CommWarrior worm implementations demonstrate the ability for an attacker to successfully infect devices in a relatively quick period of time.[30, 31, 29] Bluetooth worms pose a serious threat. To date, F-Secure has identified 71 different mobile malware worms that use Bluetooth as the vector.

2.1.4 Information Theft

Information theft attacks attempt to gain access to unauthorized information by exploiting particular security weaknesses. This section examines some information theft attacks that exist in the protocol, application profiles, and protocol layers. To illustrate the multitude of weaknesses, this section describes the encryption key weaknesses, CarWhisperer, HIDattack, BlueSnarfer, BlueBug, and iPhone MetaSploit attacks.

Encryption Key Disclosure

```
./bccmd -d hci0 psset -r bdaddr 0x75 0x00 0x12 0xB7 0x07 0x00 0x0E 0x01
```

Figure 2.1 : Modification of Bluetooth MAC Address under Linux BlueZ Stack

A weak encryption scheme plagued the Bluetooth protocol until the 2.1 release. While the 2.1 specification release provides an improved encryption protocol, nearly

2 billion devices exist with pre-2.1 specifications. Prior to the 2.1 release, a weakness existed in the pairing process that disclosed enough information to compromise the linkkey created for encryption. Tools such as Btcrack and btpincrack provide a means of implementing such attacks.[32, 33] However, both tools require the entire sequence of messages used in the pairing process. Thus, Wool forges the identity of a valid Bluetooth device and resets the pairing process.[32] Because Bluetooth does not require authentication, repudiation attacks present a problem in Bluetooth. An attacker can spoof a valid device by flashing the valid MAC address onto his Bluetooth-enabled device. Figure 2.1 shows how an attacker can simply flash a false MAC address onto a Bluetooth radio. With successful theft of a linkkey and a spoofed MAC address, a hacker can essentially hijack a device.[22]

HIDattack

The Hidattack, proposed by Mulliner, exploits the Bluetooth Human Interface Devices (HID), such as mice, keyboards or joysticks.[34] This implementation takes advantage of the flawed HID implementations. The Bluez Linux stack prior to 2.25, the Windows SP2, Widcomm, and Mac OS X stacks all fail to incorporate low-level security modes in their Bluetooth HID implementations.[34] 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 is a high threat.

CarWhisperer

The CarWhisperer implementation attacks 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.[35] 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, an attacker scans for devices that match the appropriate hands-free-audio class. Once matching the correct class,

he further checks the MAC address to determine the default passkey provided by the manufacturer. He uses the passkey to create an RFCOMM connection to the vulnerable devices. He then creates a control connection, connecting to the SCO links, which carry the audio for the Bluetooth device.[35]

BlueSnarfer

In BlueSnarfing, the attacker gains access to remote data by initiating an OBEX push.[36] 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 Aadm Laurie of the Trifinite Group discovered this vulnerability in several devices in late 2003.[36]

BlueBugger

While BlueSnarfing generally allows access to a limited number of resources on devices, a separate type of attack known as BlueBugging can cause much more significant damage. BlueBugging takes advantage of RFCOMM channels that do not require any authentication.[37] Each Bluetooth device offers 30 RFCOMM channels. In some implementations, these channels require neither authentication nor encryption. Thus, an attacker can simply connect to a vulnerable RFCOMM channel. Once connected, the attacker issues a series of commands that allow initiating calls, utilizing the SMS messaging system, updating the phonebook, call forwarding or forcing the device to utilize a certain cellular provider.[37] After gaining access, the attacker can eavesdrop without any knowledge by the attacked device. Herfurt and Laurie discovered this vulnerability and demonstrated it successfully on 50 phones during CeBit 2004.[36, 37]

HeloMoto

A HeloMoto attack targets some Motorola phones by initiating an OBEX push, but then interrupts the process without completing the OBEX push.[38] The result

places the attacker on the list of authenticated devices on the target phone. The attacker can then execute commands to the phone via the RFCOMM layer. This enables the attacker to have the same level of access as a BlueBug attack, including access to the phonebook, SMS messaging system, and phone settings.[38]

iPhone MetaSploit

Recently, Kevin Mahaffey and John Hering of Flexilis Inc. discovered a vulnerability in the Bluetooth implementation on the iPhone.[39] 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 can simplify the discovery of the Bluetooth MAC Address for the iPhone by passive capture of the WiFi traffic. Using the address captured in WiFi traffic, they can calculate the Bluetooth MAC address. As phones and mobile devices continue to grow in functionality, the potential exploits will also grow.

2.1.5 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.[28]

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.[40] 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. Means already exist to detect intrusion on WiFi and cellular protocols.[41] Thus, this thesis examines how to detect and prevent intrusion on the Bluetooth interface.

2.2 Intrusion Detection

This section reviews concepts involved in intrusion detection and applies the concepts to a Bluetooth-based intrusion detection system. First, this section examines the monitoring schemes and detection techniques of intrusion detection systems. Next, the section presents visualization schemes for intrusion detection systems. Finally, it describes an active response for wireless intrusion detection systems.

2.2.1 Network Intrusion Detection Systems

To detect malicious and suspicious traffic, security experts may employ intrusion detection systems either at the specific host or on the network. A host-based system notices only threats for that particular end point by monitoring evidence such as system calls or logs. Buennemeyer et al. developed a host-based intrusion detection system for Bluetooth battery depletion attacks that detects attacks by monitoring power levels.[28] However, a host-based system proves practical in only a limited percentage of Bluetooth devices. A smartphone or handheld computer should prompt a user with anomalous behavior as suggested by Buennemeyer.[28] However, a headset, mouse, or any peripheral device does not have the capability of displaying a message. Furthermore, most mobile devices possess limited memory and processing capabilities. Dedicating those resources to intrusion detection would limit the functionality of such devices. In contrast, a network-based intrusion detection system could provide both security and transparency to the end user. Network-based intrusion detection systems notice threats for the entire network by monitoring the flow of traffic. Thus, a Bluetooth network intrusion detection system examines the flow of traffic in Bluetooth piconets and reports anomalous or malicious behavior.

2.2.2 Anomaly-Based Intrusion Detection

Anomaly-based detection models rely on the assumption that all intrusion activities are anomalous in nature.[42] Thus, an anomaly-based detection system establishes a baseline of normal usage through extensive training and then flags anything that deviates from the established baseline as anomalous and intrusive.[43]

Anomaly-based detection models suffer problems that amplify either false positives or false negatives.[42] False positives occur when the IDS flags nonintrusive, anomalous traffic as intrusive. False negatives occur when the IDS fails to detect intrusive behavior that is not anomalous. Selection of an alert threshold level prevents unreasonable magnification of these two problems.[42] By raising the threshold level, the administrator can reduce false positives. Conversely, he can reduce false negatives by lowering the alert threshold. Additionally, anomaly-based systems require constant training to ensure the systems have an accurate representation of normal usage. Sundaram provides a survey of anomaly-based IDSs employing a statistical approach, predictive pattern generation, or usage of neural networks.[42] This section examines the use of a statistical approach and predictive pattern generation for a Bluetooth intrusion detection system.

Statistical Approach

A statistical approach relies on models such as the Bayes' theorem to identify anomalous behavior on the network.[44] As in all anomaly-based systems, a training period establishes routine normal usage. In statistical systems, the IDS compares the baseline behavior created during the training period to the current behavior. Statistical systems adaptively learn the behavior of users. Therefore, statistical systems detect intrusive behavior with a higher sensitivity than do other systems.[42] However, statistical systems present a major problem in the fact that intruders can gradually train a system to recognize intrusive behavior as baseline traffic.[42]

Although this thesis does not implement a strict statistical approach, it does provide the tools to manage traffic statistics and examine the statistics for anomalous behavior. The graphical user interface (GUI) of the Bluetooth intrusion detection sys-

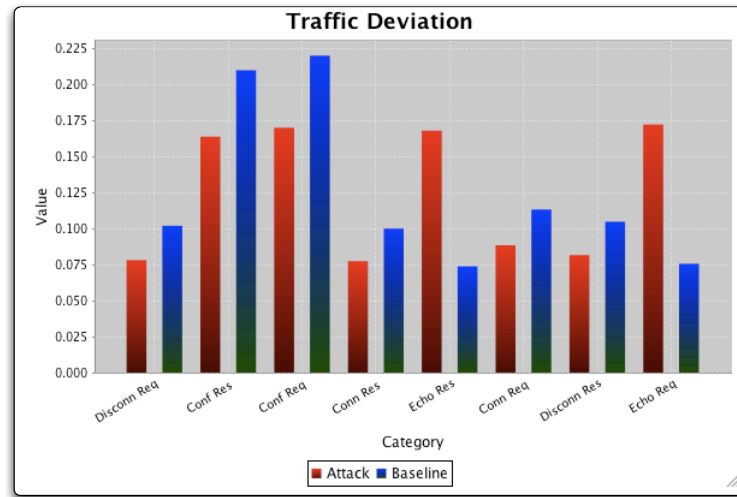


Figure 2.2 : Traffic Deviations Between Attack (BlueSmack) and Baseline Traffic

tem provides the ability to monitor several Bluetooth protocol statistics to analyze attack behavior. Figure 2.2 shows a graph generated from the GUI of the program showing the L2CAP layer Bluetooth traffic captured during a BlueSmack attack compared to previously recorded baseline traffic. A BlueSmack attack contains a higher percentage of quality of service messages (echo_req and echo_res) to implement the attack. Using this data and further analysis, a statistical approach would train the system to classify the behavior as anomalous and intrusive.

Predictive Pattern Generation

Predictive pattern generation presents another method of detecting intrusive behavior by using an anomaly detection technique. By relying on events that have already occurred, this method attempts to predict future events. The use of patterns can detect behavior unrecognizable by traditional methods and detect and report behavior faster.[42] Additionally, predictive pattern systems can detect attackers attempting to influence systems during the training period.[42]

Predictive pattern generation presents an interesting approach to detecting intrusive Bluetooth behavior. Logically, some Bluetooth attacks should follow in sequence.

For example, an information theft attack should follow the scan of a vulnerable service. Or a denial of service attack should follow discovery of devices. However, predictive pattern generation may not be able to detect an attack when the hacker has passively gained information to prepare the attack such as in the case of the iPhone MetaSploit attack.

2.2.3 Misuse Intrusion Detection

Misuse detection systems search for known signatures or patterns of unauthorized behavior.[43] While the specific signatures may be benign, they usually serve as legitimate indicators of malicious activity. In fact, most modern commercial intrusion detection systems rely on misuse detection.[45] IDSs can detect known patterns with more efficiency than attempting to detect or predict abnormal behavior.[45] Misuse detection systems may employ pattern matching, state transition analysis, or expert systems to detect intrusions.[42]

Pattern Matching

Pattern matching encodes known signatures as patterns. The IDS compares incoming traffic to the set of signatures to detect an intrusion. Simple to design and build, pattern matching systems detect known attacks. Pattern matching cannot detect novel attacks or zero-day attacks. However, pattern matching detection IDSs suffer from signature-evasion techniques. An attacker can craft an attack in such a way to avoid a weak signature.

In spite of this weakness, pattern matching often provides the most efficient and convenient means for detecting an attack. The SNORT network-based IDS provides one implementation of an IDS employing pattern matching.[46] By creating simple rules, the administrator can instruct the IDS to send an alert on malicious traffic. Further, SNORT contains a set of plug-in modules that increase the functionality of the system by performing specific analysis such as detecting a portscan attack.

State Transition Analysis

The state transition analysis model takes state design into account in order to analyze more complex attacks.[43] By maintaining states, the IDS can represent the attack scenario by using a finite state machine. Therefore, any event that leads the network from a safe state to an unsafe state proves intrusive behavior.[45]

Maintaining the state of Bluetooth packets would prove feasible. Since a Bluetooth packet is 625 microseconds long, a device can send only 1600 packets per second. Further, the maximum rate of Bluetooth ranges between 1 MB/s to 3 MB/s, based on the specification.[47] Thus, an intrusion detection system would only require caching of roughly 120 MB to 360 MB of data for a two-minute window. Further, the system can remove several packets, including the majority of radio and baseband traffic to reduce the workload required by the IDS. A modern system could easily maintain this data in physical memory without swapping.

Expert Systems

Expert system models contain a hybrid approach to intrusion detection by combining both anomaly and misuse detection models.[42] By encoding known intrusive behavior, the system can look for anomalous behavior. Expert systems attempt to overcome the constraints of misuse-only detection systems by providing the capability of detecting unclassified intrusive behavior.

Depren et al. presented a model utilizing both anomaly and misuse-based detection systems.[48] Depren implemented an IDS that contained an anomaly-based detection module and misuse-based detection module. He then correlated the output using a decision output module. The results demonstrated that the hybrid system can benefit from the strengths of both approaches, while negating the weaknesses of each approach.[48]

2.2.4 Realtime Intrusion Detection

Detecting attacks in realtime and taking countermeasures proves challenging in any intrusion detection design. However, a Bluetooth IDS requires fast notification

of intrusion because of the added mobility of the attackers. Without an immediate response, administrators are not likely to identify and capture the attackers. Thus, the IDS requirements dictate a fast notification of malicious activity.

Li et al. modified a model using the AprioriAll algorithm that correlated multiple alerts.[49] Using his algorithms, he mined attack behavior patterns and make some conclusions. Based on the current sequence of attacks, he predicted the next attack and took counter-measures. By predicting attacks, Li's model proved extremely efficient. However, it required a well-tuned behavior model prior to the attack.

In contrast, Kim et al. presented a model using a sliding window technique.[50] Kim's model proved to be an efficient event counting method for a predetermined interval. By aggregating alerts that have similar characteristics, using triggers to examine only necessary traffic and avoiding repetitive alerts, Kim's model demonstrated a less-efficient but more general purpose model.

The use of a sliding window proved to be an efficient method of examining Bluetooth traffic for malicious patterns. However, selection of a Bluetooth traffic window size proves critical. If the system selects a traffic window too small, it misses important traffic that contains data relevant to the attack. If the window is too large, the system runs inefficiently and may drop packets and not recognize malicious activity.

2.2.5 Visualization of Attacks

A major criticism of intrusion detection systems is the high rate of false positives. Displaying the large amount of information collected by a system to an administrator proves challenging. Oline and Reiners explored a means of visualizing the aggregate of intrusion detection data.[51] By examining visualized data, administrators can fine-tune intrusion detection systems and reduce the rate of false positives.

Abdullah et al. offered a visualization model that enabled a port-based overview of network activity.[52] They presented data using stacked histograms of aggregate port activity, combined with the ability to drill-down for finer details. Thus, they aided administrators in detecting attacks in realtime.

2.2.6 Active Response

Finally, this section examines the active response of the implemented Bluetooth IDS. Intrusion detection systems typically contain passive response techniques such as logging and notification. Effective response to a wireless attack could include active countermeasures.[41] Lim et al. proposed a wireless 802.11 IDS with some active countermeasures in 2003.[41] They suggested that a wireless IDS could actively respond to threats against the attacker. Further, they hypothesized that they would have success because in most cases the attackers themselves have configured their network interfaces to authenticate on the network.

This thesis implements specific responses for each category of Bluetooth attack activity. By classifying attacker behavior into a specific category, the IDS can direct a specific response at the hacker and prevent, deny, or disrupt the effects of the initial attack. In addition to the work by Lim et al., the next chapter examines related works relevant to the design of a Bluetooth IDS.

Chapter 3

Related Work

This work presents a method of detecting malicious Bluetooth traffic using misuse detection. Therefore, it incorporates a discipline of intrusion detecting and an understanding of wireless and mobile threats. While intrusion detection is a continually improving 30 year old topic, the study of wireless and mobile threats is a fairly more modern topic. This chapter reviews the related works in intrusion detection, wireless and mobile threat modeling, and current related tools.

3.1 Intrusion Detection

Anderson first described the concept of an intrusion detection system in 1980.[53] 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.[54] 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.[55] Kumar described a model of misuse detection, which encoded attacks as well defined patterns and monitors for those specific pat-

terns. 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.[56] 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.[46] 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.[46] 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.[41] Additionally, the system included the ability for an active response to wireless attackers.[41] In 2004, the US Army awarded a multi-million dollar contract to AirFortress to protect 802.11 networks in use by the military.[57]

3.2 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.[58] Biermann and Cloette examined the necessity for prevention and detection of threats against mobile devices.[59] Finally, Cache and Liu provided a comprehensive source of several wireless threat models, attacks, and implementations.[22]

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.[38, 34, 35] Additionally, A. Wool provided a means of compromising the encryption scheme of

Bluetooth.[32, 33] Haataja provided a comprehensive model for examining Bluetooth attacks.[17] Finally, the National Institute of Standards and Technology (NIST) documented several the flaws in the security design of the Bluetooth protocol.[40]

Two recent works have attempted to provide a means of modeling and detecting particular Bluetooth attacks. Yan et al. developed a model for the growth of Bluetooth works that relied upon Markov Chains.[29] 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.[28]

3.3 Current Related Tools

To capture and decode Bluetooth traffic, Spill and Bittau have provided an excellent means of recording Bluetooth traffic utilizing the Universal Software Defined Radio.[20] 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.[17]

Combs developed the Wireshark (formerly Ethereal) as a utility to capture and analyze packets.[60] 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 already 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 for example, the AirDefense BlueWatch.[24] This tool assesses the overall Bluetooth security; however, it provides no means of intrusion detection or active response. The system proposed by this thesis does both and can be incorporated with existing tools to increase the overall security of an organization. For a further understanding, the next chapter describes the design of the implemented Bluetooth IDS.

Chapter 4

Bluetooth IDS Design

4.1 Overview

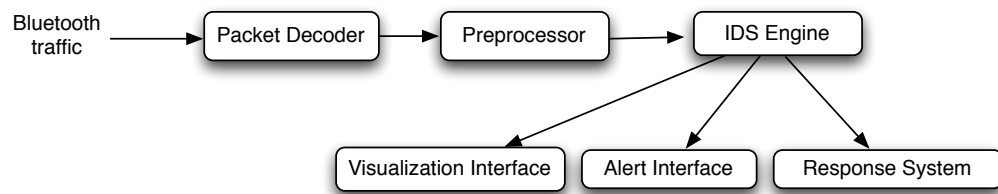


Figure 4.1 : Design of Bluetooth Intrusion Detection System

This chapter explains the design of the Bluetooth intrusion detection system implemented in this thesis. Figure 4.1 shows the arrangement of the key components in 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. The following sections present the further details regarding the key components of the system before describing the design of signatures. Finally, the last sections examine some of the reconnaissance, denial of service, and information theft attack signatures to detect particular Bluetooth attacks.

4.2 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 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.

L2CAP	Addr	C1	Packets	L2Len	L2CID	Code	Ident	SigLen	Data
135	0x1	M	2	604	Sig	Echo Req	0xDF	600	600 bytes

Packet	C1	Freq	CAC	HDR	Addr	DH5	L_CH	L2FL	Len	Data	CRC	Ack'd
121396	M	2451			0x1	0xF	UA/UI	1	339	339 bytes	0x4F87	Ack
121400	M	2457			0x1	0xF	...UA/UI	1	269	269 bytes	0xA8A8	Ack

Figure 4.2 : Packet Reassembly by the Preprocessor

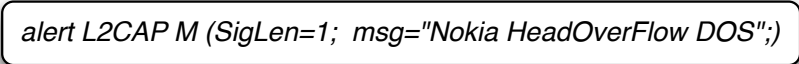
Next, the preprocessor reassembles fragmented packets and modifies the data packets to prevent signature evasion techniques. Figure 4.2 depicts the reassembly of two packet fragments back into one logical packet. In addition to reassembly, the preprocessor discards several of the baseband and radio layer 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.[46] 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 has a fixed-length sliding window determined by the time required to detect most attacks.

4.3 IDS Engine

This thesis implements a similar engine to 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.[46] The author constructed a grammar for rules included in the IDS Engine.

4.3.1 Pattern Matching



```
alert L2CAP M (SigLen=1; msg="Nokia HeadOverflow DOS");
```

Figure 4.3 : 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 4.3 demonstrates a rule to identify a Bluetooth Header Overflow Attack. The rule matches any L2CAP layer packet, originating from a Bluetooth master device, that has a Signal Length of 1 and flags it as a specific attack since the Signal Length field contains an invalid value that 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.

4.3.2 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 in this work increase extensibility of the system by allowing users to expand the number of attacks the system can detect.

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 detect the count of a particular packet type in a particular stream of traffic. Additionally, by aggregating several modules that have similar characteristics, the system increases the efficiency of the IDS engine. For further understanding of the plug-in modules implemented in this work, Figure B.5 in Appendix B provides an example of a module used to detect the Bluebugger attack.

4.3.3 IDS Engine Rule Grammar

Figure 4.4 lists the grammar for the Bluetooth Intrusion Detection System. The grammar allows the user to detect attacks either through user-specified patterns or the plug-in modules. For each rule the user can alert, log, or direct a specific response. Plug-in modules exist for attacks that require stateful inspection. Each pattern rule consists of a rule header that specifies the layer, direction bit for the rule, and a series of chain rules. Each chain rule consists of a series of rules that match specific fields of the packet. Finally, the last part of the rule is the action field, which directs the specific action for alerting, logging, or action upon matching the rule.

4.4 Visualization Interface

The visualization interface of the system provides the administrator with the ability to observe events, traffic, and alerts in a manageable manner. Visualization allows an administrator to distinguish between malicious activity and benign traffic with rel-

```

S → Rule+
Rule → PatternRule | PluginRule
PatternRule → Action+ Layer Dbit ChainRule+ ActionField+
Action → ALERT | LOG | RESP
Layer → LMP | HCI | L2CAP | SDP | RFCOMM | OBEX | APP | ANY
Dbit → M | S | ANY
ChainRule → Field = Value;
Field → L2LEN | OPCODE | CODE | L2CID | CODE | SIGLEN | DATA | TIME | ...
Value → [A-Z | 0-9 | BLANK]+
ActionField → MessageField | ResponseField
MessageField → MSG=Value;
ResponseField → RESP=Response;
Response → HONEYPOT | BREAK
PluginRule → Action+ PluginModule ActionField+
PluginModule → BLUEBUGGER | BLUESNARFER | BLUESPAM | BSS | BTCRACK |
BTSCANNER | CARWHISPERER | HELOMOTO | HIDATTACK | L2PINGOFDEATH |
PSMCAN | RFCOMMSCAN | TBEAR

```

Figure 4.4 : Bluetooth Intrusion Detection Grammar

ative ease. Within the field of intrusion detection, previous works have shown the benefit of utilizing visualization to detect attacks. [51, 52] Since the employed system does not have knowledge of zero-day attacks, a visualization tool proves beneficial in allowing the administrator to detect anomalous and potentially malicious behavior. The implemented visualization 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. Figures 4.5 and 4.6 provide examples from the graphical user interface (GUI) of the implemented system.

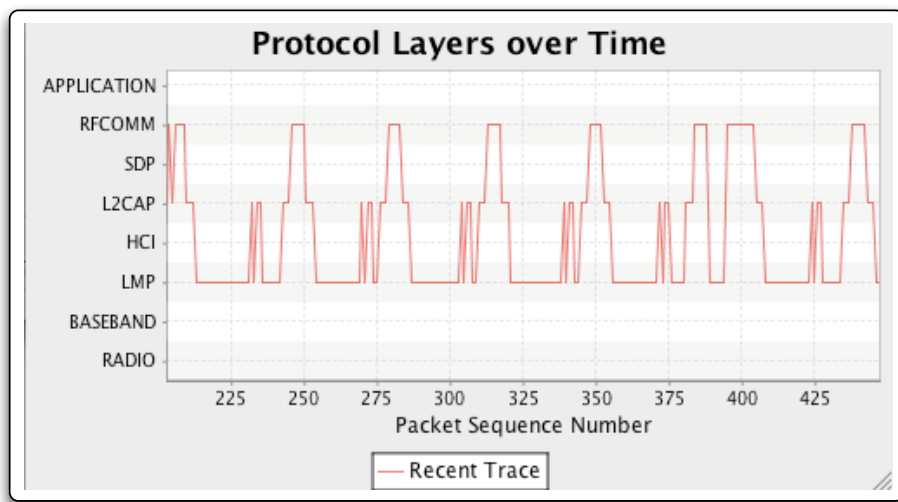


Figure 4.5 : Visualization of RFCOMM Scan Attack

Figure 4.5 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. Thus, he can drill down and look 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.

Furthermore, Figure 4.6 shows the RFCOMM channel activity of Bluetooth traffic captured during that period. In the previous 60 seconds, the attacker made 9

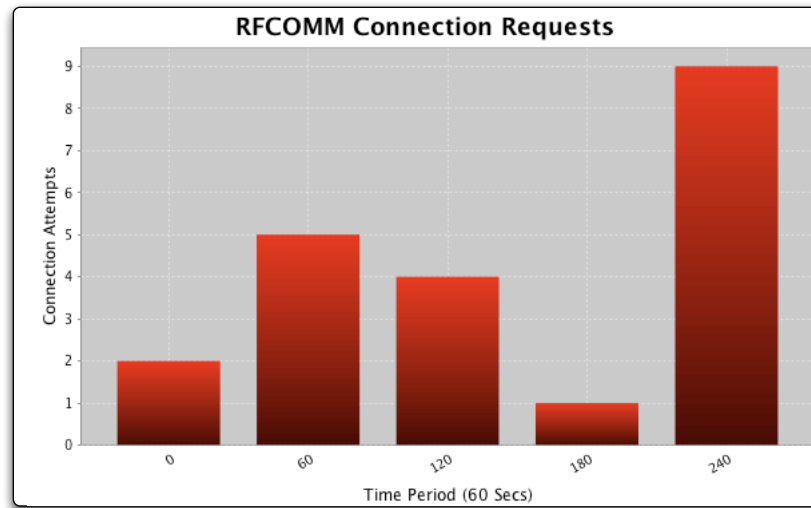


Figure 4.6 : Visualization of Bluetooth Channel Activity

RFCOMM connection attempts. Examining all of this information enables the administrator to suspect that the attacker is attempting an RFCOMM scan of a Bluetooth target. The next sections discuss the design of signatures to discover such exploits.

4.5 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. Many Bluetooth protocol stacks have been written on the assumption that the specifications would not be violated, hence the particular stacks perform poorly when handling such traffic. Appendix A.1.2 discusses the protocol stack implementations and some of the discovered vulnerabilities. Thus, the IDS signature can look for certain header fields or combinations of values to detect such attacks.

For more complex stateful inspections, the signatures must use characteristics of

an attack that are not easily evadable by an attacker. Suspicious but legitimate packets are best used in combination with other values to detect an attack. For an example, if a pattern of legitimate Bluetooth traffic causes a particular vulnerability on a particular device, then the IDS signature must take into account the intended device. Since Bluetooth devices perform feature requests that describe the capabilities of each device, the IDS has this information at its disposal. Such is the case with the HeloMoto attack that performs an attack by planting 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.

The following sections describe the design for signatures for reconnaissance, denial of service, and information theft attacks. In developing signatures for attacks, the author adhered to two rules for designing signatures.

- Signatures must be based on attack details and if possible strengthened with other characteristics to reduce the rate of false positives.
- Signatures must be written in a manner that an attacker attempting to evade signature would essentially make the attack useless.

4.6 Signatures for Reconnaissance Attacks

This section examines some signatures and plug-ins for detecting reconnaissance attacks. Detecting reconnaissance tools proves invaluable because reconnaissance likely occurs prior to denial of service or information theft attacks. This section describes the signatures for three reconnaissance attacks including Tbear, RFCOMM Scan, and the Bluetooth Stack Smasher (BSS).

4.6.1 Signature of Device Discovery Attack

First, this section examines the signature of a device discovery reconnaissance attack. Figure 4.7 depicts the flow of traffic while an attacker is employing the Tbear reconnaissance tool. To begin the attack, the hacker broadcasts an inquiry

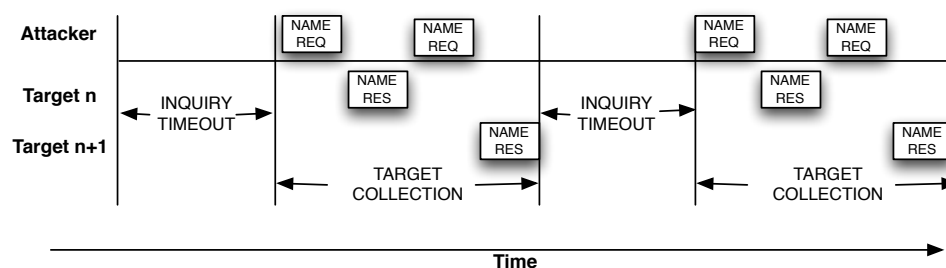


Figure 4.7 : Model for Tbear Reconnaissance Attack

request to all targets. The attacker then listens to responses from the inquiry during the inquiry timeout period. The attacker asks for each target's name by sending a Name.Request. Each target responds by sending a Name.Result. After the last target response, the attacker again broadcasts a new inquiry. The same pattern repeats itself as the attacker gathers new targets.

To discover such an attack, the IDS detects the pattern for the repeated inquiry requests, followed by repeated name requests to multiple devices. Upon detecting this repeated pattern of traffic over an extended period of time, the IDS then determines the intrusive traffic to be a device discovery reconnaissance attack. Tools such as Tbear and BTscanner implement this attack. However, each tool uses a different time to collect inquiry responses. Therefore, the system can detect both a generic device discovery attack and further classify the tool implementing the attack based on the inquiry timeout value.

4.6.2 Signature of Service Discovery Attack

Next, this section examines the signature of a service discovery attack. Figure 4.8 shows a model for an RFCOMM Scan attack. In the figure, the attacker scans a target for vulnerable RFCOMM channels by opening a sequence of connection to different RFCOMM channels over an extended period of time. Furthermore, the attacker accomplishes this attack by connecting to RFCOMM channels multiple times without requesting any type of authentication or encryption. This behavior proves

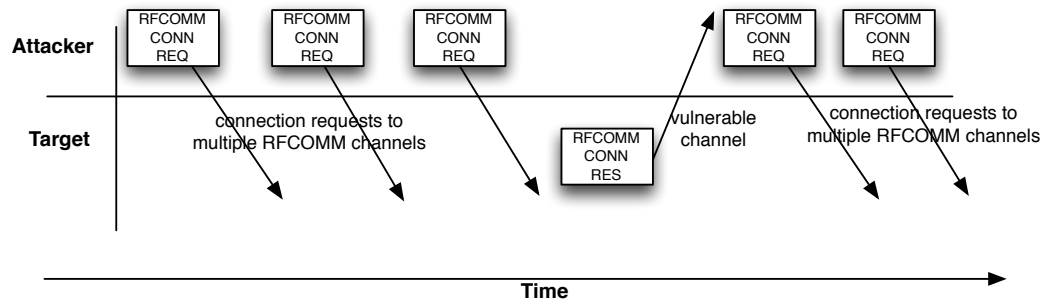


Figure 4.8 : Model for a RFCOMM Scan Reconnaissance Attack

similar to a TCP portscan where an attacker may attempt to connect to open TCP ports. Additionally, when the attack succeeds, the target responds with an RFCOMM.CONN.RES message indicating that the target allowed the attacker access to the channel without authentication or encryption.

Thus, the pattern matches when an attacker repeatedly attempts to connect to multiple RFCOMM channels without requesting any authentication or encryption. A similar attack, PSM Scan, demonstrates the same behavior. However, PSM Scan makes repeated attempts to connect to the PSM services without requesting any authentication or encryption. Thus, the IDS detects the PSM scan in a similar manner by looking for repeated connections to PSM services without any prior authentication or encryption.

4.6.3 Signature of Exploit Discovery Attack

This section examines the signature of an exploit discovery attack. Figure 4.9 shows a captured Bluetooth Stack Smasher Attack that generated random L2CAP packets in order to discover vulnerabilities on a target. The hexi-decimal signaling codes for the L2CAP packets do not translate to logical commands because they have been specially crafted by the attack. Thus, the signature for a BSS attack includes a set of L2CAP packets with malformed signaling codes.

L2CAP	Addr	C1	Packets	L2Len	L2CID	Code	Ident	SigLen	Data	Time
208	0x1	M	5	1661	Sig	0xB3	0x24	22948	1657 bytes	101.614s
209	0x1	M	5	1481	Sig	0xB3	0x7B	31431	1477 bytes	102.118s
210	0x1	M	3	934	Sig	0x75	0x5A	36733	930 bytes	102.623s
211	0x1	M	5	1376	Sig	0x8E	0x50	51995	1372 bytes	103.128s
212	0x1	M	11	3232	Sig	0xFA	0x44	23515	3228 bytes	103.634s

↑ Out-of-range L2CAP signal codes used to detect exploits
 ↑ randomized data used for fuzzing devices

Figure 4.9 : Bluetooth Stack Smasher Reconnaissance Attack

4.7 Detection of Denial of Service Attacks

This section explains the implementations of some of the signatures for denial of service attacks. Specifically, this section examines the BlueSpam, Nasty vCard, Tanya, Nokia N70, and Header Overflow attacks.

4.7.1 Signature of BlueSpamming

First, this section examines the signature of the BlueSpam tool. The BlueSpam tool repeatedly sends unsolicited traffic to targets discovered through the built-in discovery features of Bluetooth. In a BlueSpam attack, the hacker repeatedly connects to a target and initiates an unsolicited object transfer. To detect such an attack, the IDS looks for repeated OBEX transfers and detects the behavior as a BlueSpam attack.

In some instances, the attacker can craft and send objects with particular vulnerabilities. Thus, the IDS contains a rule for any unsolicited object transfer for particular objects. Figure 4.10 shows a malicious payload sent by an attacker. The particular vulnerability causes a buffer overflow error on certain devices by using a

To detect the use of these tools, the IDS looks for the specially crafted quality of service messages. The Tanya attack uses a special packet depicted by Figure 4.11 . The packet has an additional out-of-range signaling command with a data field of 4068 bytes. This extra signaling command causes problems in decoding the packet on some devices.

Further, to differentiate between the attacks, the IDS looks at the size of the data field of the messages for Tanya, Symbian Remote Restart, and BlueSmack attacks. Each tool uses different size quality of service messages including 20, 44, and 667 bytes respectively. The different size messages cause vulnerabilities on different vendor devices.

4.7.3 Signatures of Maliciously Formed Bluetooth Packets

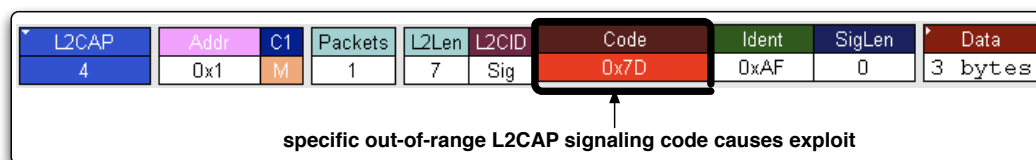


Figure 4.12 : Nokia N70 Denial of Service Attack

To detect maliciously formed bluetooth packets, the IDS uses patterns that match the malicious fields. Figure 4.12 shows an example of a malformed packet that causes denial of service on certain Nokia N70 mobile devices. By sending an L2CAP packet with an code of 7D, an attacker can crash certain Nokia N70 mobile device. 7D does not indicate a reserved code and is therefore not a valid field. When an attacker sends such a packet to a Nokia N70 Device, the IDS matches the known signature and detects the traffic as a specific Nokia N70 attack.

Figure 4.13 shows a second malicious packet that causes an attack. By setting the Signal Length to an invalid value greater than 12, an attacker forces a vulnerable target to reference an invalid memory location. Thus, the IDS rule looks for any

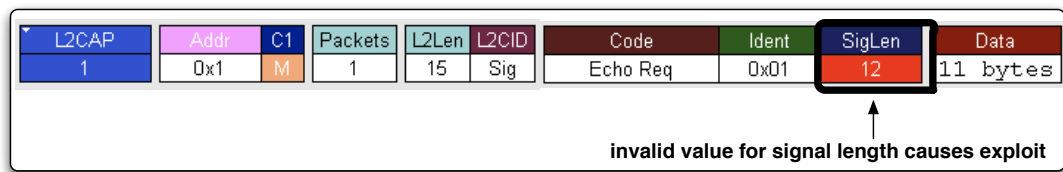


Figure 4.13 : L2CAP Header Overflow Denial of Service Attack

L2CAP packet with a malformed Signal Length and also flags it as a specific L2CAP HeaderOverflow Attack. Furthermore, the IDS detects a similar attack known as HCIDump-Crash that uses a different invalid value for the Signal Length that forces a crash on some versions of the Linux BlueZ protocol stack.

4.8 Detection of Information Theft Attacks

This section explains signatures used in the testbed to detect information theft attacks. Specifically, it looks at the signatures for the Btcrack, CarWhisperer, BlueBugger, HIDattack, and Helomoto attacks.

4.8.1 Signature of Btcrack

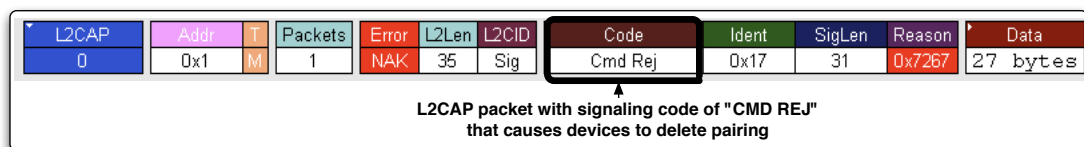


Figure 4.14 : Capture of a Packet Used to Reset and a Connection for BTCrack

First, this section examines a signature to detect hackers using the Btcrack tool. To observe the pairing process and discover the key used for encryption, attackers send a specially crafted message, known as a Btcrack, to disrupt the connection. Figure

4.14 shows the capture of the specially crafted packet. The signature for a Btcrack attack involves detecting this L2CAP CMD_REJ packet followed by immediate pairing and mutual authentication of both Bluetooth devices. Such behavior allows an attack access to enough information to execute the Btcrack attack to compromise the linkkey.

4.8.2 Signature of CarWhisperer Attack

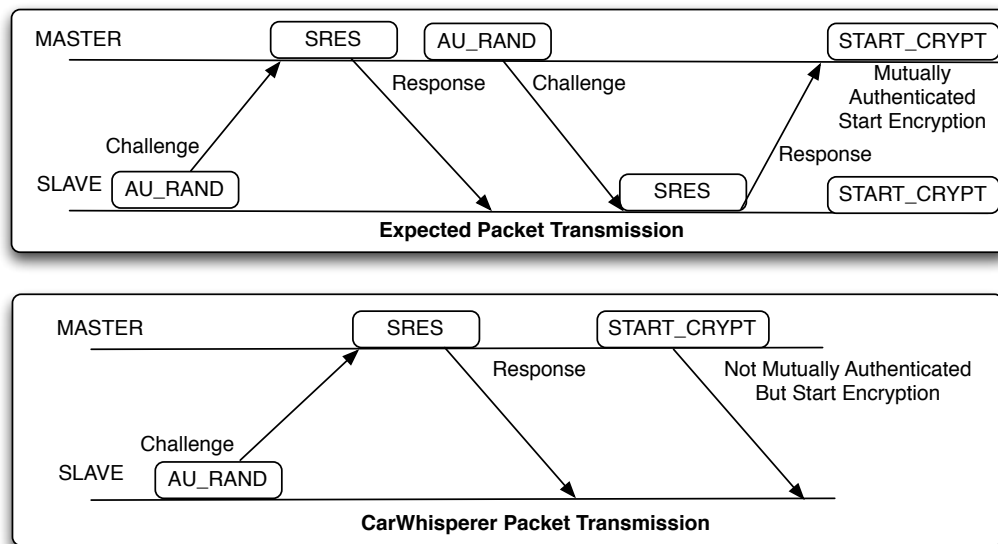


Figure 4.15 : CarWhisperer Packet Transmission

Figure 4.15 demonstrates the behavior during a CarWhisperer attack compared to normal Bluetooth behavior. According to the protocol specification, both sides should perform mutual authentication by challenging with an AU_RAND message and calculating an SRES response. Following mutual authentication by both parties, both sides request that future transmission be encrypted with the START_CRYPT message. During the CarWhisperer attack, the attacker challenges the target with an AU_RAND and the target replies back with a calculated SRES response. However,

the target does not authenticate the attacker. The attacker then initiates encryption without performing authentication on the target. After the traffic is encrypted, the attacker connects to the SCO channel and begins receiving the audio links. Therefore, the signature for a CarWhisperer attack includes an incomplete authentication followed by immediate encryption and encrypted SCO traffic.

4.8.3 Signature of BlueBug Attack

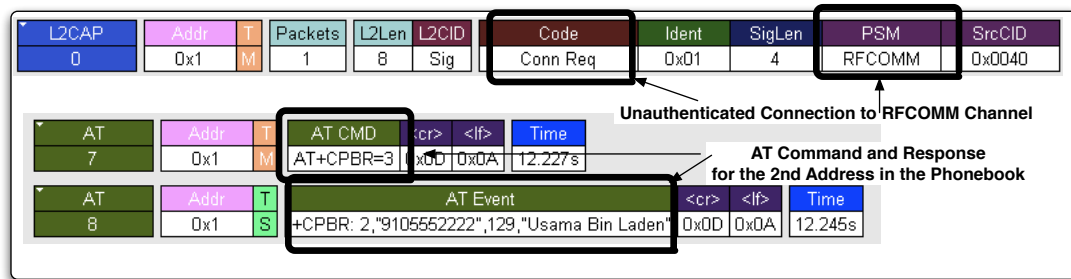


Figure 4.16 : BlueBug Attack Traffic Captured by IDS

The signature of a BlueBug attack involves detecting an unauthenticated or unencrypted RFCOMM connection to a mobile device, followed by RFCOMM-AT commands that result in the theft of information from the device. Figure 4.16 demonstrates a BlueBug attack. First, the attacker connects to an RFCOMM channel without authentication. Next, the attacker issues a request for an address in the phonebook. The unsuspecting target device responds with the information.

Unfortunately many manufacturers allow the connection to RFCOMM channels without previous authentication or encryption. While this activity itself poses a security risk, it is the actual theft of information that causes the IDS to raise an alert and flag the behavior as intrusive. Furthermore, the IDS also detects BlueSnarf attacks in the same manner. The BlueSnarf tool implements the attack in the same manner but uses different AT commands to access the device. As such, the IDS differentiates between the two different tools on the basis of the issued AT commands.

4.8.4 Signature of HeloMoto attack

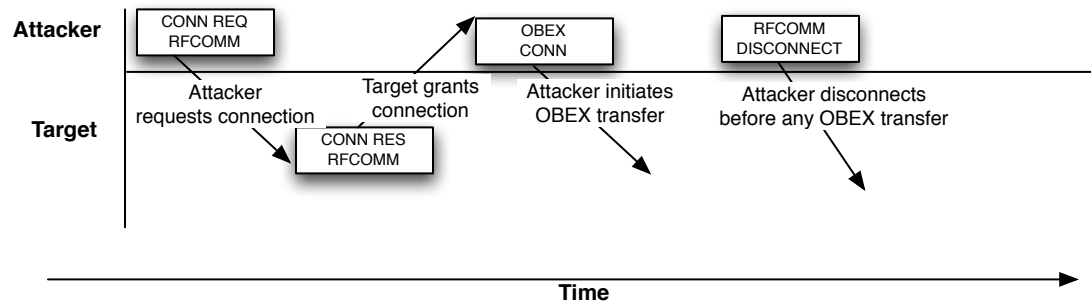


Figure 4.17 : Pattern for HeloMoto Traffic

The HeloMoto attack grants access to an attacker on certain Motorola devices. In order to implement the attack, the hacker initiates an RFCOMM connection to a vulnerable device. The hacker then requests to transfer a file via OBEX. However, the attacker then terminates the connection without completing the OBEX transfer. This uncompleted transfer places the hacker as an authorized user on the vulnerable device. Figure 4.17 depicts the the attack. Thus, to detect the attack, the IDS looks for RFCOMM connections to a mobile phone where an OBEX transfer was requested but not attempted or completed prior to a request to disconnect the connection.

4.8.5 Signature of HIDattack

Finally, this section examines the signature pattern for the HIDattack tool used to implement attacks against Bluetooth HID devices. The HIDattack gives an attacker the ability to connect to some Bluetooth-enabled computers to take over control of devices such as the wireless keyboard and mouse. This attack takes advantage of the fact that the protocol does not enforce authentication or encryption. To execute this attack, a hacker simply connects to the HIDP interrupt and control channels of the target and then has access to falsely act as the HID keyboard or mouse. Figure 4.18 demonstrates the traffic pattern for such an attack. Thus, the IDS simply defines an

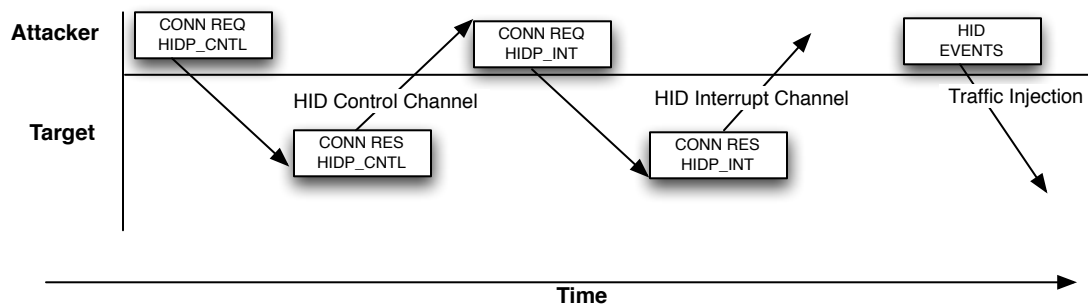


Figure 4.18 : Pattern for HIDattack Traffic

unauthenticated connection to the HIDP control and interrupt Channels followed by a series of HID events. A separate method of implementing the HIDattack relies upon passive capture of Bluetooth traffic. As such, the implemented intrusion detection system cannot detect this. The next section also discusses some of the problems faced in developing the system.

4.9 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 close scrutiny 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.

4.9.1 Reconnaissance Response

For responding to a reconnaissance attack, this thesis presents a method of deploying decoys or honeypots.[61] By standing up honeypot targets, the system distracts attackers from more valuable machines on the network.[61] 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 thesis implements the response system on a separate machine. By separating the response system from the IDS engine, the author demonstrates the ability to deploy multiple separate response nodes.

The Bluetooth IDS implementation employs a production honeypot system. 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, the system creates a mirage of several active Bluetooth devices to distract the attacker from the legitimate Bluetooth devices.

4.9.2 Denial of Service Response

Next, this work implements a means of responding to denial of service attacks 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.[32] However, this work uses a similarly crafted message to terminate an attacker. Upon immediate discovery of a denial of service attack, the implemented 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.

4.9.3 Information Theft Response

Finally, this work implements a response to an information theft attack. Upon detection of an information theft attack, the response node stands up a false target device with the same physical address as the vulnerable device. By doing so, the system provides a phony target device with similar services as the vulnerable device to distract the attacker. Thus, the IDS prevents the attacker from connecting to any

vulnerable targets to perform future attacks until the administrator can physically locate the threat. To demonstrate the success of such a method, the next chapter describes the implementation of the IDS and intrusion responses for this work.

Chapter 5

Bluetooth IDS Implementation

5.1 Overview

This chapter provides an explanation of the implementation and testing of the Bluetooth intrusion detection system. First, this chapter examines the testbed used to record the metrics of the system. Next, this chapter presents some of the practical problems and limitations faced when constructing the first network-based Bluetooth intrusion detection system.

5.2 Bluetooth IDS Testbed

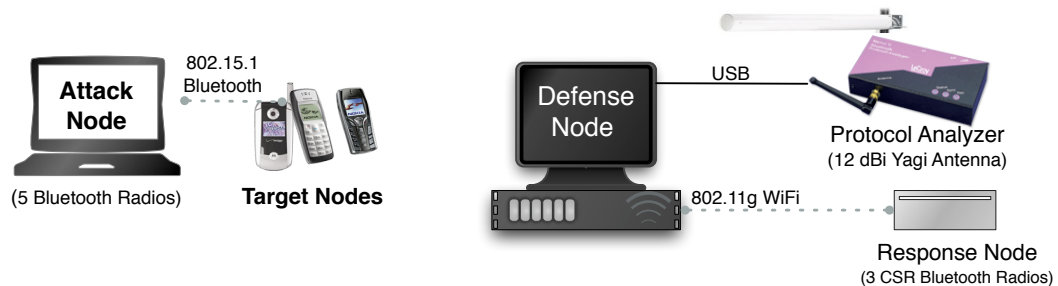


Figure 5.1 : Testbed Used for Bluetooth Intrusion Detection

The testbed for this work included a defense node, response node, attack node, and vulnerable target nodes. Figure 5.1 provides a graphical representation of the experiment. The defense node maintained the responsibility of 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.

5.2.1 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 in Section 2.2. 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 author wrote several small reconnaissance programs to test the timing and frequency of differing reconnaissance probes for discoverable and non-discoverable devices. Appendix C includes version information for all tools used on the attack node. Table 5.1 lists the complete set of attacks launched at the system to test the ability to evaluate the implemented IDS.

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.

Table 5.1: Table of Bluetooth-Enabled Attacks

Attack	Category	Description
BlueBugger	Information Theft	Implements the BlueBug attack to steal mobile device data
BlueSnarfer	Information Theft	Implements the BlueSnarf attack to steal mobile device data
Btcrack	Information Theft	Resets the encryption key between two devices
HIDattack	Information Theft	Impersonates HID Bluetooth devices
HeloMoto	Information Theft	Steals access to a mobile device on particular Motorola phones
CarWhisperer	Information Theft	Injects audio into and records audio from some hands-free audio devices
BlueSpam	Denial of Service	Repeatedly sends unsolicited Bluetooth content
Nasty vCard	Denial of Service	A vCard that causes a denial of service in some Bluetooth devices
HCIDump-Crash	Denial of Service	A denial of service attack by a maliciously crafted L2CAP packet
L2CAP HeaderOverflow	Denial of Service	A denial of service attack by a maliciously crafted L2CAP packet
BlueSmack	Denial of Service	A denial of service attack that sends large L2CAP Echo Requests
Ping of Death	Denial of Service	A denial of service attack via L2Ping
Tanya	Denial of Service	A denial of service attack via maliciously crafted L2Ping packets
Nokia N70 DOS	Denial of Service	A denial of service attack that affects some Nokia devices
Symbian Remote Restart	Denial of Service	A DoS attack that affects some Symbian devices
Tbear	Reconnaissance	A device discovery tool for discoverable Bluetooth devices
BTScanner	Reconnaissance	A device discovery tool for discoverable Bluetooth devices
RFCOMM Scan	Reconnaissance	A service discovery tool (part of the BTAudit package)
PSM Scan	Reconnaissance	A service discovery tool (part of the BTAudit package)
Bluetooth Stack Smasher	Reconnaissance	An exploit discovery tool that fuzzes the L2CAP layer

5.2.2 Target Nodes

Table 5.2: Targeted Devices

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

The targets in the experiment included a variety of devices with well documented manufacturer flaws. 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 5.2 outlines some attacks and the respective target devices.

5.2.3 Defense Node

The defense node consisted of a hardware protocol analyzer and a software IDS application. The Merlin LeCroy Protocol Analyzer proved to be an excellent solution for recording Bluetooth traffic because it can nonintrusively capture, display, and analyze Bluetooth piconet data. Additionally, the Merlin Analyzer included support for addressing the device via a scripting language. The scripting language for the device enabled the author to listen to devices on a specific frequency or specific piconet. Further, it allowed recording and logging of traffic on of 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.

```
alert BLUEBUG; msg=Bluebugger Attack;
alert BLUESNARFER; msg=BlueSnarfer Attack;
log L2CAP any code=CMD REJ; msg=Specific Btcrack Message Sent;
alert BTCRACK; msg=Btcrack Attack;
alert HIDATTACK; msg=HIDAttack;
alert HELOMOTO; msg=Helomoto;
alert CARWHISPERER; msg=CarWhisperer Attack;
alert BLUESPAM; msg=BlueSpam Attack;
alert L2CAP any code=echo req; Siglen=12; msg=HCIDumpCrash Attack;
alert OBEX any Opcode=Fput; Data=186 bytes; Msg=Nasty vCard Detected (186 bytes);
alert L2CAP any Siglen=1; msg=Sony Ericsson DOS (L2CAP HeaderOverflow Attack);
log L2CAP any code=echo req; data=600 bytes; msg=BlueSmack Attack (L2Ping of 600 Bytes);
log L2CAP any code=echo req; data=667 bytes; msg=BlueSmack Attack (L2Ping of 667 Bytes);
alert BLUESMACK; msg=Bluesmack Attack;
alert PINGOFDEATH; msg=Ping of Death Attack;
log L2CAP any code=echo req; code=unknown; Data=20 bytes; Msg=Tanya Single Packet Attack;
alert TANYA; msg=Tanya Denial of Service Attack;
alert L2CAP any Code=Unknown; SigLen=0; Data=3 bytes; Msg=Nokia N70 DOS (Echo Request with SigLen=0 && Data = 3 Bytes);
alert L2CAP any Code=Echo Req; data=44 bytes; msg=Symbian Remote Restart (Also Default Size L2Ping Packet);
alert LMP any name length=248 bytes; name fragment=???; msg=BackTrax Attack Tool in Use;
alert LMP any Opcode=name_res; name fragment=phone of doom.; msg=BlueDiving Tool in Use;
alert TBEAR; msg=Tbear Recon Detected;
alert BTSCANNER; msg=BTScanner Recon Detected;
alert RFCOMMSCAN; msg=RFCOMM Scan;
alert PSMSCAN; msg=PSM Scan;
alert BSS; msg=Bluetooth Stack Smashing;
```

Figure 5.2 : Default Rule List

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 design of a Bluetooth IDS engine outlined in Chapter 4. Additionally, the application included a graphical interface that provided the alert and visualization interfaces. Furthermore, the application provided the security administrator with the capability to further configure rules and write add-on modules for future attack signatures. Appendix B.1 provides more information about the use of the software application.

5.2.4 Intrusion Response Node

To demonstrate the response capabilities hypothesized earlier in this thesis, the author constructed an intrusion response node with the responsibility 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.

5.3 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.

5.3.1 Protocol Analyzer Packet Decoding

Next, the hardware analyzer presented a problem in realtime testing of three specific attacks, because of the version of the analyzer. For the research, the author utilized an older Merlin CATC LeCroy Protocol Analyzer that can decode Bluetooth traffic compliant with the 1.1 Core Specification. Later versions of the Analyzer, including the Merlin II, include support for the succeeding Bluetooth Core Specification. The older analyzer presented a problem as the analyzer required additional specifications to decode a limited set of packets. Specifically, the analyzer required specific channel decoding assignments to understand which protocol layer had generated RF-COMM traffic for some packets during the BlueSnarfer, BlueBugger, and HeloMoto attacks. Thus, the author could only record these attacks manually, because they required the additional input to the analyzer. The author tested the remainder of the attacks autonomously utilizing the scripting language of the analyzers.

5.3.2 Scheduling Between Bluetooth Piconets

Additionally, the protocol analyzer presented a problem in the fact that it cannot 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 could simultaneously listen to 79 different frequencies and then reorganize each captured packet into a group for a particular piconet. Such a system would require 79 unique radios and a multiplexing scheme. Separately, a user could attempt to synchronize with a unique piconet and hop in sequence with that particular piconet. Alternating between piconets provides an efficient, but not optimal, picture of traffic. The possibility does exist that one could miss an attacker during the swapping of piconets. However, in the long term, the system would likely identify an aggressive attacker. The testbed for this work employed a system that scheduled unique piconets for recording in a round-robin algorithm. Appendix B.2 provides an example script to record data in different piconets and upload the data to the IDS software application.

Chapter 6

Evaluation of Results

6.1 Overview

This chapter evaluates the implemented intrusion detection system in terms of the Dense Advanced Research Projects Agency (DARPA) metrics for intrusion detections and the implemented response capability.

6.2 Experiment Setup

The author derived the results in this chapter using the testbed explained in the previous chapter. To test the probability of detection, the author generated traffic on a testbed network as suggested by Mell et. al. [62] Using 20 different attack tools, the attack node ran exploits against the series of vulnerable target nodes. The author then recorded each attack with a LeCroy Bluetooth Protocol Analyzer. Furthermore, the author built a special analyzer recording options file to reduce the amount of traffic uploaded to the IDS. To increase the efficiency of the system, the analyzer dropped all baseband and radio traffic. Additionally, the author specified decoding assignments for RFCOMM AT traffic that the protocol analyzer could not autonomously decode. The author then verified each signature with off-line recordings.

Next, the author tested 17 different attacks with on-line detection by streaming the data from the protocol analyzer to the IDS preprocessor. During the on-line

detection tests, the author controlled the analyzer via a scripting language and forced it to autonomously upload decoded packets to the IDS preprocessor while the attack node performed a series of attacks on the targets. During this time, the analyzer only recorded data from the defense node instead of round robin scheduling.

To verify the probability of false alarms, the author used previously recorded benign traffic from the LeCroy Corporation, consisting of 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 then examined the entire collection of packets in less than 30 seconds.

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

6.3 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.[62]

This paper utilizes 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.

6.3.1 Coverage

Coverage defines the attacks that an intrusion detection system can detect under ideal conditions.[62] For signature-based systems, coverage defines the set of known, defined signatures. The coverage of the implemented system consists of 20 known attacks listed in Table 5.1 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.

6.3.2 Response Time

Table 6.1: Time Required for Detection of Attacks

Attack	Category	Avg. Required TWin
RFCOMM Scan (5 Conn. Requests)	Reconnaissance	110.86 sec
PSM Scan (100 Conn. Requests)	Reconnaissance	4.747 sec
HeaderOverFlow	Denial of Service	0.0006 sec
Nasty vCard	Denial of Service	1.1030 sec
BlueSnarf	Information Theft	1.4696 sec
BlueBugger	Information Theft	3.2566 sec
CarWhisperer	Information Theft	0.2277 sec
HeloMoto	Information Theft	3.2294 sec

Figure 6.1 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 author 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.

6.3.3 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.[62] To test the probability of detection, the author attacked the target nodes with all 20 attacks listed in 5.1. The IDS used the previously calculated sliding window of 120 seconds of traffic. Furthermore, the author 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.

6.3.4 Probability of False Alarms

False alarms incorrectly produce alerts on benign background traffic.[62] To test the results of the system implemented here, the LeCroy corporation provided a baseline of off-line recordings of 33 different devices, including Bluetooth-enabled computers, headsets, phones, HID devices, and handheld computers. The recordings consisted of a total of 26,031 logical packets of benign Bluetooth traffic. The author ran the Bluetooth IDS with the offline recordings from the LeCroy Corporation to determine a rate of false positives. The system did not produce any false-negative alerts on any of the benign traffic.

6.3.5 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.[62] All of these attacks attempt to overwhelm the IDS with data in order to decrease its ability to make an intelligent decision. The limited throughput of Bluetooth decreases the ability 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 the Bluetooth packets to logical data. Fragmented packets are reassembled by the protocol analyzer with transparency 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 must simultaneously record and multiplex all 79 frequencies used by the Bluetooth protocol.

6.3.6 Ability to Identify an Attack

The ability to identify an attack depicts the accuracy that an IDS can identify an attack with a specific common name or vulnerability name. [62] 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.

6.3.7 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 author 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 invulnerable device and identified the attack as unsuccessful.

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 6.1 shows the GUI alert. The author then repeated the same experiment with an invulnerable

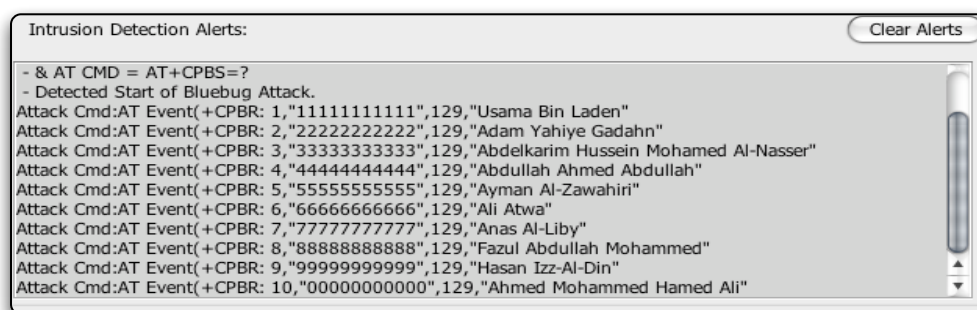


Figure 6.1 : Report of Data Stolen During BlueBug Attack

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.

6.3.8 Processing Speed

Table 6.2: Off-line Processing Time

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

One measurement for determining the relative responsiveness of an intrusion detection system is the speed at which the IDS can process traffic. Table 6.2 shows traffic processed off-line by the IDS and 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.

6.4 Evaluation of Response Capability

Finally, this section reports the results of testing the distraction and disruption responses employed to prevent reconnaissance, denial of service, and information theft attacks. Through the usage of flash-enabled Bluetooth radios, the system created honeypots to distract attackers and sent falsified messages to terminate attacks.

6.4.1 Reconnaissance Response

Honeypots have been used effectively to distract attackers from IP targets.[61] 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 in this thesis implemented the following system. 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 must flash 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 author 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.

6.4.2 Denial of Service Responses

The author further tested the IDS response system by attempting to break ongoing denial of service attacks using falsified connection termination messages.[32] 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 scanned for all discoverable devices. The response system then forged packets from all of those devices to the attacker.

```
0 bytes from 00:60:57:F1:97:AA id 32 time 51.07ms
0 bytes from 00:60:57:F1:97:AA id 33 time 39.07ms
0 bytes from 00:60:57:F1:97:AA id 34 time 44.21ms
0 bytes from 00:60:57:F1:97:AA id 35 time 47.04ms
0 bytes from 00:60:57:F1:97:AA id 36 time 42.16ms
0 bytes from 00:60:57:F1:97:AA id 37 time 76.25ms
0 bytes from 00:60:57:F1:97:AA id 38 time 48.23ms
Send failed: Connection timed out
```

Figure 6.2 : Attacker's Console after Denial of Service Response

First, the system 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. Figure 6.2 depicts the results on the attacker's

console. Further testing showed that the response capability disrupted similar attacks such as Tanya, Ping of Death, and Symbian Remote Restart attacks.

6.4.3 Information Theft Responses

Next, the author 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.



```
Shell - Blueprint <2>
attack-box / # sudo bluebugger info -a 00:60:57:F1:97:AA
bluebugger 0.1 ( MaJoMu | www.codito.de )
-----
Target Device: '00:60:57:F1:97:AA'
Target Name: 'Phony-Target-Response'
RFCOMMCREATEDDEV failed: Success
Cannot open '/dev/rfcomm1': Connection refused
attack-box / #
```

Figure 6.3 : Attacker's Console after Information Theft Response

To test the defense, the author simply flashed a radio with the same physical address as a vulnerable device. The author 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. Figure 6.3 shows the results of the failed attack. 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.

Chapter 7

Conclusion

7.1 Summary

1. Bluetooth is emerging as 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.
2. 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.
3. This thesis implements a network intrusion detection system, based on a misuse detection scheme, to detect Bluetooth traffic. This system contains the same constraints of typical misuse detection schemes such as the lack of ability to detect new attacks. However, the system provides an efficient and effective means of detecting intrusive attacks. Furthermore, the system demonstrates

the ability to determine the success of an attack and shows moderate resistance to IDS attacks.

4. Finally, 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 ceasing ongoing attacks through specially crafted messages.

7.2 Limitations and Future Work

7.2.1 Hybrid Model for Detection

The system presented in this thesis suffers from the same constraints of all misuse detection engines. It cannot detect zero-day or unclassified attack behavior. Rather, the system relies upon the system designer to develop and provide signatures for known attacks. Future work could examine the usage of an anomaly-based intrusion detection such as statistical analysis to detect anomalous and intrusive behavior. Additionally, a hybrid model could potentially be utilized to find anomalous behavior.

7.2.2 Correlation of Multiple Bluetooth Sensors

The implemented IDS processes input from a single protocol analyzer. Thus, it allows the IDS to observe behavior in only a single 1 km location. Further, the system records unique piconets. A future system could take advantage of multiple protocol analyzers to process data from several piconets in different locations. Similarly, a future system could deploy 79 radios to listen to traffic on each unique Bluetooth frequency in a given location. A future system could employ a specific custom firmware for Bluetooth dongles. A specific firmware could improve the packet decoding and preprocessing.

7.2.3 Further Intrusion Response Development

This thesis presents some responses to denial of service attacks. In most cases, end users will be able to determine a denial of service attack occurred. However, this thesis develops signatures in order to determine the originating MAC address of the attack. Work by Rodriguez et al. demonstrates how to determine the location of a Bluetooth device based on the MAC address and received signal strength indicator (RSSI).[63] Thus, a future system could ultimately identify the physical location of the attacker. To defend against denial of service attacks, the IDS could target the attacker with a denial of service attack.

The IDS could simply frequency jam the attacker by transmitting on the preamble of each frequency during the hopping period. In either case, it would reduce the ability of the attacker to perform a denial of service attack against targets while the hacker spent time defending himself. Directing such a response requires careful scrutiny so as not to disrupt the traffic of other users. A future work could examine the response methods in greater detail.

This work has shown the ability to stand up false targets with the same physical address as vulnerable devices. A future work could expand upon this by deploying a firewall or screen to protect vulnerable devices when a hacker begins attacking devices. The firewall could act as a relay by intercepting Bluetooth traffic and replaying only benign traffic to and from the vulnerable devices.

7.2.4 Integration with Existing Defense Tools

This thesis presents an intrusion detection tool for the Bluetooth. Currently tools exist to assess Bluetooth security in organizations by querying specific device profiles, services, and channels. A future work could examine integrating an intrusion detection tool into such Bluetooth security assessment tools. Using the knowledge of weak or vulnerable devices, the IDS could direct specific responses to protect such devices from attack.

7.2.5 Integration with Existing Intrusion Detection Systems

Furthermore, a future work could integrate a Bluetooth IDS into an existing IDS such as SNORT. Such a work would have to modify the packet capture library, packet decoder, preprocessor, and detection engine components of the existing IDS. SNORT, for example, uses a packet capture library that processes Data-Link (Layer 2 of the OSI Model) packets. In order to integrate both IDSs, the system would require a translation of Bluetooth packets to an OSI Layer 2 Form. The Wireshark Packet Capture Tool available at wireshark.org has already begun working on integrating and translating Bluetooth H4 HCI packets into a libpcap format.[60] A future work could expand this work as SNORT uses the same packet capture library as Wireshark.

7.3 Speculation

Finally, this thesis examines two potential ways in which Bluetooth-enabled attacks may progress in the near future, including attacks against the newest protocol specification and attacks against more-critical targets in industry.

7.3.1 Attacks Against the 2.1 Protocol

The Bluetooth 2.1 Core Specification brings significant security improvements. Most notably, the use of the Diffie-Helman Key Exchange prevents passive eavesdropping attacks from overhearing enough information to crack the linkkey used for encryption. However, the Bluetooth 2.1 Core Specification ensures that 2.1 Core devices remain compatible with previous devices that suffer from weaker encryption schemes. The potential exists that an attacker may be able to rollback the encryption scheme by crafting messages that make the communicating devices appear as 2.0 and below devices instead of 2.1 devices. As such, the attacker can gain access to the encryption key. Attackers have demonstrated similar rollback attacks on the SSL protocol with success.

7.3.2 Attacks Against Infrastructure

Because of the lack of a current security infrastructure to protect Bluetooth, it remains only a matter of time before a significant attack occurs via Bluetooth. Imagine a fictional attack. Because of the inherent discovery and object transfer functionalities included in Bluetooth, the potential to mass-market Trojan horses exists. Outside of the New York Stock Exchange, a Bluetooth-enabled attacker could discover and distribute specially crafted images to unsuspecting targets. The images would simply appear as advertisements for the latest coffee franchise but actually run executable code by exploiting a buffer overflow in an image-handling library. The code would instruct the device to upload the file contents of the device to a remote server for later inspection. Or the code could instruct the device to infect other devices via another protocol. Within a short time and with relative ease, the hacker would be able to capture large amounts of sensitive financial and personal data.

Bibliography

- [1] Installed base of more than one billion products gives consumers more than five billion ways to use the global wireless standard. Technical report, The Bluetooth Special Interest Group (SIG), Nov 2006.
- [2] Fiona Thomson. Bluetooth enabled equipment shipments to hit 800 million this year. Technical report, IMS Research, Oct 2007.
- [3] Bluetooth SIG. Core specification v2.0 + EDR. Technical report, Bluetooth SIG, Nov 2004.
- [4] David Cypher, Nicolas Chevrollier, Nicolas Montavont, and Nada Golmie. Prevailing over wires in healthcare environments: Benefits and challenges. *Communications Magazine, IEEE*, 44(4):56–63, Apr 2006.
- [5] Daniel Beaumont. Bluetooth brings mobility to health care. *Planet Wireless*, pages 11–15, 2002.
- [6] David Miller. Mobile finance: Pay as you go. *Unwired Magazine*, 5:16–17, 2007.
- [7] Mexican bank deploys hypercom bluetooth-enabled payment stations. *Mobile Enterprise Magazine*, Oct 2007.
- [8] Bob Brewin. AirDefense sniffs out Bank of America Bluetooth-based ID system. *Computer World*, May 2004.
- [9] Kate Kaye. Navy campaign takes Bluetooth plunge. *ClickZ News*, 2007.

- [10] DoD wireless push email system security requirements matrix. Technical Report 2.0, DISA Field Security Operations (FSO), Jun 2007.
- [11] BAA 07-46 Proposer Information Pamphlet (PIP): Landroids. Technical report, Defense Advanced Research Projects Agency (DARPA), Jul 2007.
- [12] Owen Holland, John Woods, Renzo DeNardi, and Adrian Clark. Beyond swarm intelligence: the ultraswarm. In *IEEE Swarm Intelligence Symposium SIS2005*, pages 217–224, Jun 2005.
- [13] H.R. Everett, E.B. 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] John Hering, James Burgess, Kevin Mahaffey, Mike Outmesguine, and Martin Herfurt. Long distance snarf. http://trifinite.org/trifinite_stuff_lds.html.
- [15] Mike Outmesguine. Bluetooth a mile away. *Popular Science*, Nov 2004.
- [16] Ryan Woodings, Derek Joos, Trevor Clifton, and Charles D. Kutson. Rapid heterogeneous ad hoc connection establishment: accelerating bluetooth inquiry using IrDA. In *Wireless Communications and Networking Conference (WCNC2002)*, pages 342–349, Mar 2002.
- [17] Keijo 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)*, volume 7, pages 63–103, 2005.
- [18] Alex van Es. Bluetooth tracking. <http://www.bluetoothtracking.org/>, Dec 2007.
- [19] Miska Repo. Going around with bluetooth in full safety. Technical report, F-SECURE, 2006.

- [20] Dominic Spill and Andrea 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.
- [21] Ollie Whitehouse. War nibbling: Bluetooth insecurity. Technical report, @stake, 2003.
- [22] Johnny Cache and Vincent Liu. *Hacking Wireless Exposed*. McGraw-Hill, 2007.
- [23] Collin Mulliner. Bt audit. http://trifinite.org/trifinite_stuff_btaudit.html.
- [24] Enterprise wireless monitoring for bluetooth networks (AirDefense BlueWatch). Technical report, Air Defense Inc, 2005.
- [25] Pierre Betoin. Insecruite du bluetooth de nouvelles menaces. Technical report, Security Labs, Feb 2006.
- [26] United States Computer Emergency Readiness Team. libpng denial of service vulnerability. <https://www.kb.cert.org/vuls/id/684664>, May 2007.
- [27] Adam Laurie, Marcel Holtmann, and Martin Herfurt. Bluesmack. http://trifinite.org/trifinite_stuff_bluesmack.html.
- [28] Timothy Buennemeyer, Theresa Nelson, Michael Gora, Randy Marchany, and Josephy Trong. Battery polling and trace determination for bluetooth attack detection in mobile devices. In *Information Assurance and Security Workshop, 2007. IAW '07. IEEE SMC*, pages 135–142, Jun 2007.
- [29] Guanhua Yan, Leticia Cuellar, Stephan Eidenbenz, Hector D. Flores, Nicolas Hengartner, and Vicent Vu. Bluetooth worm propagation: Mobility pattern matters! In *2nd ACM symposium on information, computer and communications security*, pages 32–44. ACM, 2007.
- [30] Jing Su, Kelvin Chan, Adrew Miklas, Kenneth Po, Ali Akhavan, Stegan Saroiu, Eyal de Lara, and Ashvin Goel. A preliminary investigation of worm infections

- in a bluetooth environment. In *4th ACM workshop on recurring malware*, pages 9–16. ACM, 2006.
- [31] Guanhua Yan and Stegan Eidenbenz. Bluetooth worms: Models, dynamics, and defense implications. In *Computer Security Applications Conference (ACSAC '06)*, pages 245–256. Discrete Simulation Sci. (CCS-5), Dec 2006.
- [32] Yaniv Shaked and Avishai Wool. Cracking the bluetooth pin. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 39–50. ACM, Jun 2005.
- [33] 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)*, pages 365–373, Sep 2005.
- [34] Collin Mulliner. HID attack (attacking HID host implementation). <http://www.mulliner.org/bluetooth/hidattack.php>.
- [35] Martin Herfurt. Carwhisperer. http://trifinite.org/trifinite_stuff_carwhisperer.html.
- [36] Martin Herfurt. Bluesnarfing @ CeBIT 2004: Detecting and attacking bluetooth-enabled cellphones at the Hannover Fairground. In *Salzburg Research Forschungsgesellschaft*, pages 1–12, Mar 2004.
- [37] Adam Laurie, Marcel Holtmann, and Martin Herfurt. Bluebug. http://trifinite.org/trifinite_stuff_bluebug.html.
- [38] Adam Laurie. Helomoto attack. http://trifinite.org/trifinite_stuff_helomoto.html.
- [39] Kevin Mahaffey and John Hering. <http://www.flexilis.com/>.
- [40] Tom Karygiannis. Wireless network security: 802.11, bluetooth and handheld devices. Technical report, NIST, Nov 2002.
- [41] 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*, pages 68–75. IEEE, Jun 2003.

- [42] Aurobindo Sundaram. An introduction to intrusion detection. In *Crossroads*, volume 2 of *Special issue on computer security*, pages 3–7. ACM, Apr 1996.
- [43] Julia Allen, Alan Christie, William Fithen, John McHugh, Jed Pickel, and Ed Stoner. State of the practice of intrusion detection technologies. Technical report, Networked Systems Survivability Program, 2000.
- [44] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Bayesian event classification for intrusion detection. *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 14–23, Dec 2003.
- [45] Alfredo Serrano. Integrating alerts from multiple homogeneous intrusion detection systems (under the direction of Dr. Peng Ning.). Master’s thesis, North Carolina State University, 2003.
- [46] Martin Roesch. SNORT - lightweight intrusion detection for networks. In *13th LISA Conference*, pages 229–238, Nov 1999.
- [47] Core specification v2.1 + EDR. Technical report, Bluetooth Special Interests Group (SIG), Aug 2007.
- [48] Ozgur Depren, Murat Topallar, Emin Anarim, and M Ciliz. An intelligent intrusion detection system (ids) for anomaly and misuse detection in computer networks. In *Expert Systems with Applications*, volume 29, pages 713–722. Bogazici University, Electrical and Electronics Engineering Department, Information and Communications Security (BUICS) Lab, Bebek, Istanbul, Turkey, Elsevier Ltd, Jun 2005.
- [49] Zhitang Li, Aifang Zhang, Jie Lei, and Li Wang. Real-time correlation of network security alerts. In *e-Business Engineering (ICEBE 2007)*, pages 73–80. IEEE, Oct 2007.
- [50] Jinoh Kim, Koohong Kang, JungChan Na, Ikkyun Kim, Kiyoun Kim, Jongsoo Jang, and Sungwon Sohn. Practical network attack situation analysis using

- sliding window cache scheme. In *9th Asia-Pacific Conference on Communications (IEEE Cat. No.03EX732)*, volume 3. IEEE, Sep 2003.
- [51] A. Oline and D. Reiners. Exploring three-dimensional visualization for intrusion detection. In *Visualization for Computer Security, 2005. (VizSEC 05)*., pages 113–120, Oct 2005.
- [52] Kulsoom 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*, pages 100–108, Jun 2005.
- [53] J.P. Anderson. Computer security threat monitoring and surveillance. Technical report, James P. Anderson Co., 1980.
- [54] D.E. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2):222–232, February 1987.
- [55] Sandeep Kumar and Eugene H. Spafford. A Pattern Matching Model for Misuse Intrusion Detection. In *Proceedings of the 17th National Computer Security Conference*, pages 11–21, 1994.
- [56] Anup K. Ghosh and Aaron Schwartzbard. A study in using neural networks for anomaly and misuse detection. In *SSYM'99: Proceedings of the 8th conference on USENIX Security Symposium*, pages 12–12, Berkeley, CA, USA, 1999. USENIX Association.
- [57] Fortress dominates wireless security market protecting over 10,000 networks. <http://www.fortresstech.com>, December 2004.
- [58] Donal Welch. Wireless security threat taxonomy. In *2003 IEEE Workshop on Information Assurance*, United States Military Academy, West Point, NY, Jun 2003.
- [59] Elmarie Bierman and Elsabe 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, volume 30, pages 141–148, 2002.

- [60] Angela Orebaugh, Gilbert Ramirez, Josh Burke, and Larry Pesce. *Wireshark & Ethereal Network Protocol Analyzer Toolkit (Jay Beale's Open Source Security)*. Syngress Publishing, 2006.
- [61] Iyatiti Mokube and Michele Adams. Honeypots: concepts, approaches, and challenges. In *Proceedings of the 45th annual southeast regional conference*, pages 321–326. SIGAPP: ACM Special Interest Group on Applied Computing, ACM, 2007.
- [62] P. Mell, V. Hu, R. Lipmann, J. Haines, and M. Zissman. An overview of issues in testing intrusion detection systems. Technical report, National Institute of Standards and Technology, 2007.
- [63] Miguel Rodriguez, Juan Pece, and Carlos J. Escudero. In-building location using bluetooth. In *International Workshop on Wireless Ad-hoc Networks (IWANN'05)*, May 2005.
- [64] Nikos Mavrogiannopoulos. On bluetooth security. Technical report, Dec 2005.
- [65] Bluetooth device access guide. Technical report, Apple Inc., Dec 2007.
- [66] Simple pairing whitepaper. Technical Report Release Version V10r00, Bluetooth Special Interests Group (SIG), Core Specification Working Group, Aug 2006.

Appendices

Appendix A

Bluetooth Primer

A.1 Bluetooth

A.1.1 Overview

For purposes of understanding the specific threats to Bluetooth devices, this thesis reviews the design and specifications of the Bluetooth protocol. The Bluetooth Special Interest Group (SIG) provides the core specification documents for the protocol.[47, 3] This review specifically addresses the frequency spectrum, protocol layers, profiles, methods for discovering devices, pairing, and security modes of the Bluetooth protocol.

Frequency-Hopping Scheme

Similar to the 802.11 wireless protocols, the Bluetooth protocol utilizes the unlicensed 2.4 GHz frequency band. Unlike the IEEE 802.11 protocols that use a fixed frequency in the 2.4 GHz band, Bluetooth utilizes frequency-hopping to avoid interference with other devices operating at 2.4 GHz.[47] Communicating devices frequency hop between 79 unique frequencies from 2.400 to 2.4835 GHz.[47] However, frequency-hopping provides only minimal security because the communicating device broadcasts the frequency-hopping sequence in an unencrypted format.[64] Rather than employing security at the physical layer, the designers included the security modes in the

upper protocol layers and application profiles.

Piconet Establishment

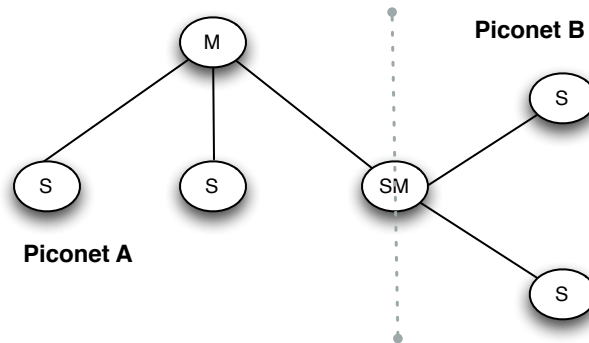


Figure A.1 : Example of a Bluetooth Scatternet

A piconet is an ad-hoc collection of connected Bluetooth devices. Figure A.1 shows an example of two Bluetooth piconets. Each piconet consists of one master device and a maximum of seven active slave devices. Thus, a 3 bit active member address (AMA) represents each distinct device in the piconet. A device may serve as a master in one piconet and a slave in another piconet at the same time. Multiple piconets are known as a scatternet. The intrusion detection system proposed in this thesis records and analyzes traffic from distinct piconets, searching for signatures of malicious traffic.

The master device periodically synchronizes devices in the piconet by broadcasting the clock offset and frequency-hopping sequence. The master broadcasts this information in a frequency-hopping synchronization (FHS) control packet that also contains the physical medium access control (MAC) address, forward error correction and a CRC code.

In order to detect existing piconets, a device must receive the broadcasted FHS packet. Once synchronized with the hopping sequence, a device may initiate a connection to the master device. While it is possible to detect a Bluetooth device and

piconet by overhearing transmitting packets, the intrusion detection system implemented in this thesis relies on the fact that the master device is transmitting FHS control packets.[20]

Packet Format

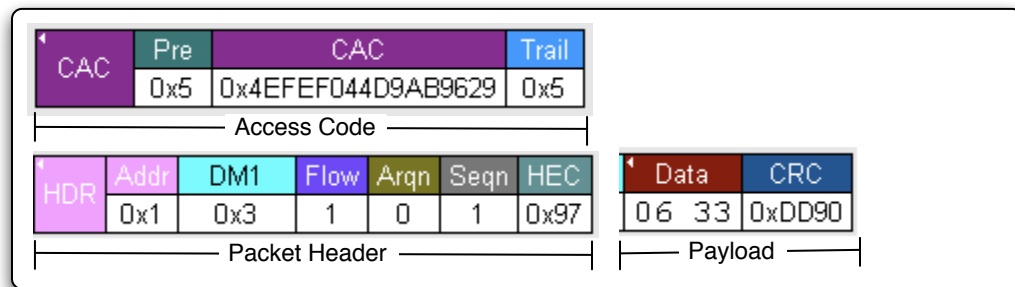


Figure A.2 : Example for a Bluetooth Baseband Packet

To explain the use of captured packets in the proposed intrusion detection system, this section reviews the format of a Bluetooth baseband packet. The packet format consists of three distinct parts: the access code, packet header, and data payload.

Figure A.2 shows an example of a Bluetooth baseband packet. The 72 bit access code consists of a channel access code (CAC), that identifies the piconet, a device access code for paging requests and response, and an inquiry access code to discover devices. The 54 bit packet header consists of a member address, type code, flow control, acknowledgment, sequence number, and header error check. The data payload is up to 2745 bits.

Bluetooth uses two types of connection links: the asynchronous connectionless link (ACL) and the synchronous connection-oriented (SCO) link. ACL packets provide data communication. In contrast, SCO links typically contain time-bounded traffic, such as audio transmissions. An ACL packet may consist of one, three, or five time slots and further carry a data-high (DH) or data-medium (DM) rating. In the next section, this thesis examines how the different layers handle typical Bluetooth packets.

A.1.2 Protocol Stack

This section examines the Bluetooth protocol stack. First, this section reviews the specific protocol layers and their specific functionalities. Next, this section examines why different protocol stack implementations have enabled some Bluetooth attacks.

Stack Design

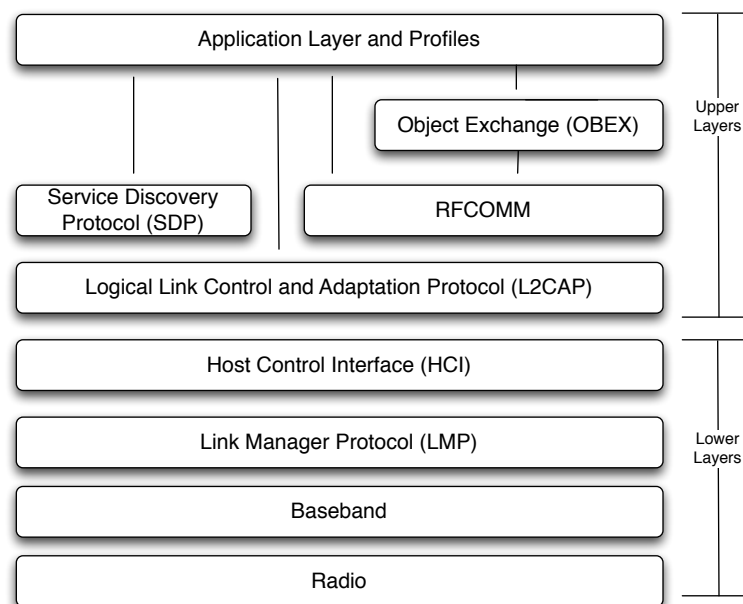


Figure A.3 : The Bluetooth Protocol Stack

Figure A.3 shows a model of the Bluetooth protocol layers. At the bottom of the stack, the Radio Layer handles modulation and demodulation of data into radio frequency signals.[65] Above that, the Baseband Layer allows Bluetooth devices to operate in three different classes based on power constraints and distance requirements. Figure A.4 depicts the three classes.[47] According to the design, devices can communicate up to a range of 100 meters.[47] However, commercial products provide the means to communicate over 30km. This increased range presents some interesting methods for attacking Bluetooth presented in Section 2.2.1.

Next, the Link Manager Protocol (LMP) translates Baseband layer operations into host commands.[65] Link Manager secures and configures the link of the established Bluetooth network. The Host Controller Interface (HCI) provides a means of accessing the Baseband controller and Link Manager. Above the HCI Layer exists the upper layers of the Bluetooth protocol stack. The Logical Link Control and Adaptation Protocol (L2CAP) Layer multiplexes the data for higher layer protocols, handles data segmentation, and manages quality of service and transmission management.[64, 47] The Service Discovery Protocol (SDP) handles discovering services on a Bluetooth-enabled device. The RFCOMM layer handles Radio Frequency emulation of serial communication. Lastly, the Object Exchange (OBEX) layer provides a means for exchanging data objects. While the specification provides an overview for how to implement the stack, each vendor provides a slightly different implementation.

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

Figure A.4 : Link Manager Protocol (LMP) Classes of Bluetooth Power

Stack Implementations

Several different vendor implementations of the Bluetooth stack exist including the Linux based BlueZ stack, Windows based Widcomm stack, and the MAC OS X Bluetooth stack. Hackers have discovered and exploited security weaknesses in several of the Bluetooth stacks. Table A.1 lists some of the reported vulnerabilities. Chapter 3 of this thesis provides a more in-depth look at these vulnerabilities.

Table A.1: Bluetooth Protocol Stack Vulnerabilities

Stack	Discovered Vulnerabilities
BlueZ Stack	denial of service attack
Toshiba Bluetooth Stack	buffer overflow, directory traversal
Widcomm	remote audio eavesdropping, remote code execution
Sony Ericcson	denial of service attacks
MAC OS X	remote code execution, worms

A.1.3 Application Profiles

At the Application layer, the Bluetooth protocol defines generic application profiles. Profiles provide specifications in order to ensure device compatibility between different vendors. Profiles include methods for distributing audio, networking, exchanging objects and accessing data. Abstracting profiles away from specific manufacturers ensures compatibility between different vendor devices.[47]

The Object Push Profile (OPP) serves as one example of a profile. The OPP defines the requirements for exchange of data objects, typically business cards, over Bluetooth. Because it does not require prior authentication on some vendor implementations, the OPP proves to be a vulnerability in some Bluetooth-enabled devices.[36]

Other profile implementations such as the Human Interface Device (HID) and hands-free-audio, contain weaknesses in certain vendor implementations. Section 2.2.4 reviews particular attacks on the HID and hands-free-audio profile implementations on some machines..

A.1.4 Device Discovery

Physical Address

The Bluetooth Medium Access Control (MAC) address, a pivotal piece of information to an attacker, identifies a unique device. Figure A.5 shows an example of a MAC address. The 48-bit address consists of the 16-bit Non-Significant Address Part (NAP), 8-bit Upper Address Part (UAP), and 24-bit Lower Address Part (LAP).[20] The first 24 bits of the address, including the NAP and UAP, identify unique vendors.

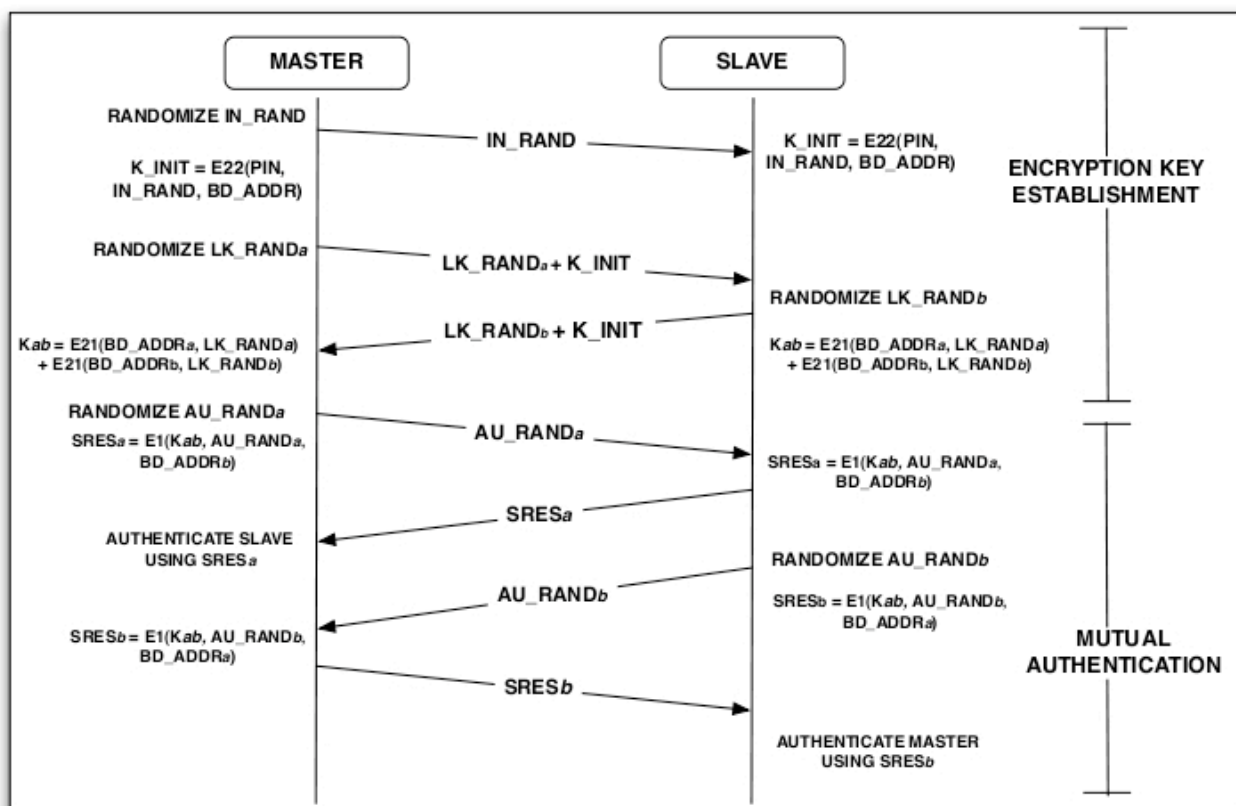


Figure A.6 : Pairing and Mutual Authentication Prior to Core Specification 2.1

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.[47] 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 linkkeys. Following creation of the linkkeys, each device pair perform mutual authentication. Figure A.6 depicts the pairing and authentication in the previous specification. Should an attacker be able to observe the pairing process, he can reconstruct the linkkeys to decrypt further traffic between paired devices.[32, 33]

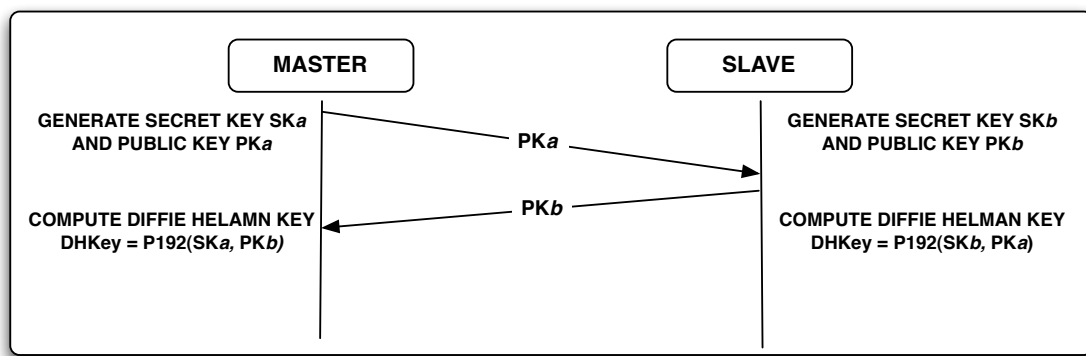


Figure A.7 : Creation of Diffie Helman Key used for Secure Simple Pairing in Core Specification 2.1

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.[47] Figure A.7 shows how the protocol creates a Diffie Helman key for authentication use. 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 re-created 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.[66] 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. The conclusion of this thesis makes speculations about attacks on NFC cooperation and potential roll back attacks.

A.1.6 Security

Security Modes

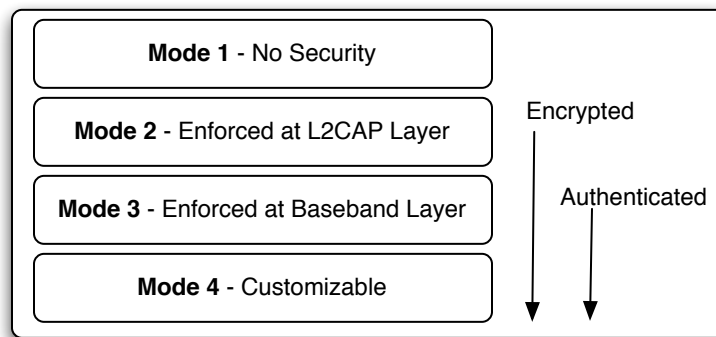


Figure A.8 : Link Manager Protocol Security Modes

While pairing provides the linkkey used for encryption and authentication, the Link Manager Protocol (LMP) directs the security mode. Four modes exist for Bluetooth security.[47] Figure A.8 depicts these modes. 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.[66, 47, 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.

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 as a nist.gov.[40] 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.[40] 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.[40] 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.

Appendix B

User's Manual

B.1 Intrusion Detection Graphical User Interface

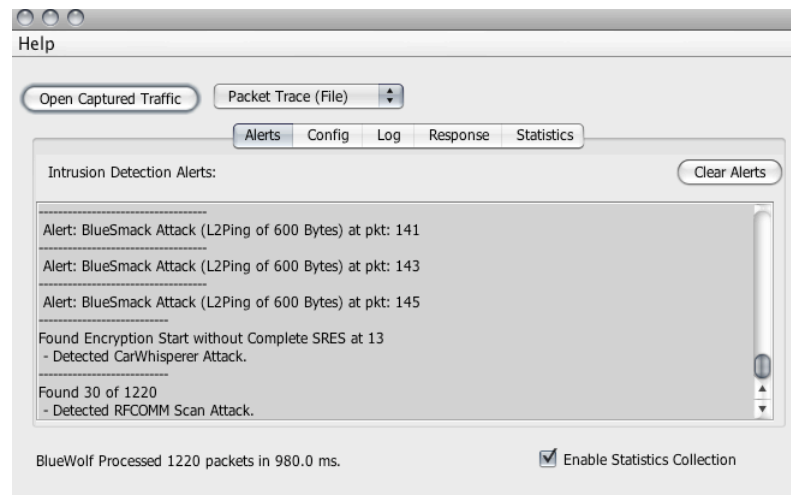


Figure B.1 : Graphical User Interface

The Bluetooth Intrusion Detection System contains a graphical user interface. Figure B.1 presents the interface. Using the interface, the administrator may select between the alert, configuration, logging, response and statistics windows. Furthermore, the administrator may choose to import traffic either from a captured trace file in packet-view format or by importing a live streaming feed of traffic. The status

bar at the bottom of the application provides the details of the current actions of the application. Further, the user may wish to disable statistics collection to increase the overall performance and efficiency of the application.

B.1.1 Rule Configuration

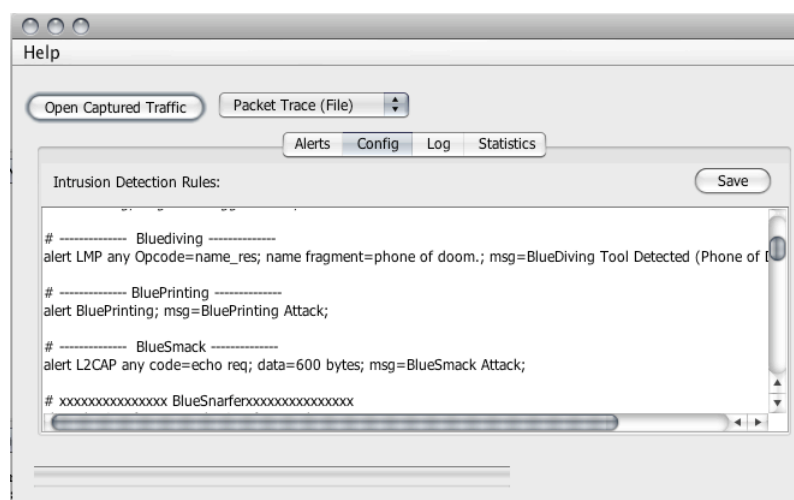


Figure B.2 : Graphical User Interface

The configuration settings allow the user to develop new signatures and write specific rules to capture packets. Figure B.2 depicts the configuration interface. Further, specific plug-in modules to detect specific attacks exist and the user may simply just wish to make specific calls to these modules. These modules can detect more complex attacks such as BlueSnarfer, BlueBugger, and CarWhisperer.

B.1.2 Response Actions

The response interface allows the user to implement the response capabilities, including the ability to deploy false targets, deny service, and terminate the connection of an attacker. A separate application, `responsenode.jar` must be running on a Linux based machine to implement these capabilities and the be specified in the response node text window.



Figure B.3 : Graphical User Interface

B.1.3 Statistics Analysis

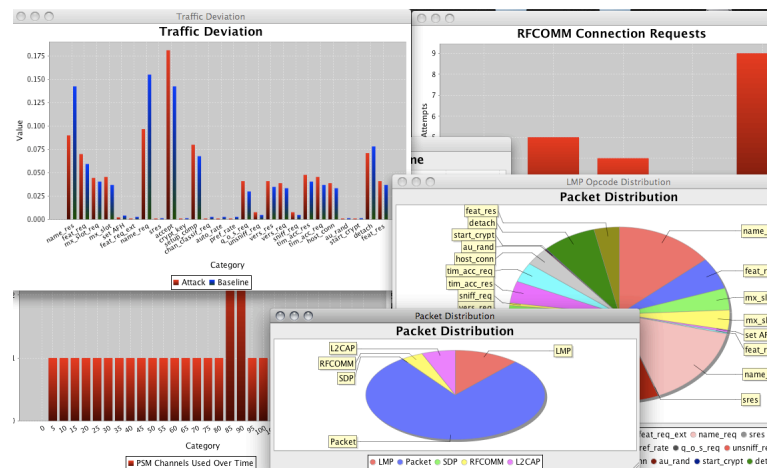


Figure B.4 : Graphical User Interface

The statistics interface allows the user to examine the current trace and previous recordings to detect anomalies and statistics in the traffic. The user has a wide variety of statistical options that will generate graphs for the data requested. Figure B.4 provides some of the graphs generated by the graphical user interface.

```

public boolean BlueBugger() {
    boolean ENCRYPT_FOUND = false; // Encryption Found in PktList
    boolean FOUND = false; // Encryption Not Found & Connection Request
    boolean DETECTED = false; // Detected Bluebugger
    String Attack = new String(); // Attack Commands

    int pkt_cnt = 0;
    // If Encryption Not Found
    for (PktProc.pktType p : pktL.L) {
        if ((match(p, "Opcode", "in_rand")) ||
            (match(p, "Opcode", "au_rand"))) {
            ENCRYPT_FOUND = true;
        }
        // If RFCOMM Connection Not Found
        if ((match(p, "Code", "Conn Req")) &&
            (match(p, "PSM", "RFCOMM"))) {

            if (ENCRYPT_FOUND == false) {
                FOUND = true;
            }
        }
        pkt_cnt++;
        // If FOUND and AT CMD To Open Phone Book Exists
        if (FOUND && match(p, "AT CMD", "AT+CPBS=?")) {
            if (pkt_cnt > idx) {
                DETECTED = true;
            }
        }
        // If Commands Found, Add to CMD List
        if (FOUND && p.Layer.contains("AT")) {
            for (PktProc.fieldType fd : p.FieldList) {
                if (fd.Field.equalsIgnoreCase("AT Event(")) {
                    Attack = Attack + '\n' + "Attack Cmd:" + fd.Value;
                }
            }
        }
    }
    // If Detected --> Output to MainText
    if (DETECTED) {
        GP.mTextArea.append("\n-----");
        GP.mTextArea.append("\n - Detected Unauthenticated RFCOMM Conn");
        GP.mTextArea.append("\n - & AT CMD = AT+CPBS=?");
        GP.mTextArea.append("\n - Detected Start of Bluebug Attack.");
        GP.mTextArea.append(Attack);
    }
    return DETECTED;
}

```

Figure B.5 : Example of a Bluetooth Module

B.2 Example of a Bluetooth Module

Figure B.5 provides an example Bluetooth plug-in module. The example module detects a BlueBugger attack. All the modules have the same format and the system allows future modules to be created and integrated into the system.

B.3 Script for Analyzer Control

The following code is a script to instruct the protocol analyzer to record, export, and upload data to the IDS engine.

```
Set Analyzer = WScript.CreateObject("CATC.Merlin")  
Set Neighborhood = Analyzer.GetBTNeighborhood  
For Each Device In Neighborhood  
Set Trace = Analyzer.MakeRecording ("")  
Trace.ExportToText "recording.txt"  
Dim objSHE : Set objSHE = CreateObject("WScript.Shell")  
Return = objSHE.Run("java -jar StreamClient.jar 192.168.105.2 recording.txt  
Next
```

Appendix C

Bluetooth Attack Information

Software utilized in this testing included Merlin Software 2.01, Lecroy Automations API version 1.40, BackTrack 2.0, Bluebugger 0.1, Bluediving 0.8, Bluesnarfer 0.1, Bluetooth Stack Smasher 0.8, BTAudit 0.1., BTCrack, BTCrawler, BTScanner 2.1-3, Cabir-Worm-F, CarWhisperer 0.2, GreenPlaque 1.4G, Ghattotooth, HCIDump Crash, Helomoto, HidAttack 0.1, PSM Scan 0.1.1, RFCOMM Scan 0.1.1, Tanya 1.5, Tbsearch 1.5, L2CAP Packet Gen 0.8, BCCMD.

The Bluetooth attack software used in the testing came from the following websites:

<http://trifinite.org/>

<http://bluetooth-pentest.narod.ru/>

<http://bluediving.sourceforge.net/>