OPTIMIZATIONS OF BATTERY-BASED INTRUSION PROTECTION SYSTEMS

Theresa Michelle Nelson

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

Master of Science
In
Computer Engineering

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

April 29, 2008

Blacksburg, VA

Keywords: Intrusion Detection System, Mobile Device Security.

OPTIMIZATION OF BATTERY-BASED INTRUSION PROTECTION SYSTEMS

Theresa Michelle Nelson

Dr. Joseph Tront, Committee Chair
Bradley Department of Electrical and Computer Engineering

ABSTRACT

As time progresses, small mobile devices become more prevalent for both personal and industrial use, providing malicious network users with new and exciting venues for security exploits. Standard security applications, such as Norton Antivirus and MacAfee, require computing power, memory space, and operating system complexity that are not present in small mobile devices. Recently, the Battery-Sensing Intrusion Protection System (B-SIPS) was devised as a means to correct the inability of small mobile devices to protect themselves against network attacks. The B-SIPS application uses smart battery data in conjunction with process and network information to determine whether the mobile device is experiencing a battery depletion attack. Additionally, B-SIPS provides mobile device statistics to system administrators such that they can analyze the state of the wireless network more thoroughly. The research presented in this thesis collaborates with and extends the B-SIPS research through optimizations and validation. Areas of focus include ensuring public acceptance of the application through the implementation of a usability study and verifying that the deployment of the application will not jeopardize the performance of external mobile device applications. Additionally, this thesis describes how GUI optimizations are realized for both the B-SIPS client and CIDE server, how future smart battery hardware implementations are introduced for increased effectiveness with the B-SIPS application, and it discusses how an optimum deployment data transmission period is determined.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Joseph Tront, for his guidance throughout the duration of my graduate studies here at Virginia Tech, and for his patience with both myself and my research during periods of research setbacks.

I would also like to thank Mr. Randy Marchany for mentoring me for the last four years, introducing me to the world of research, as well as the world of Network Security, and for always lending a listening ear when I needed one. Additionally, I would like to thank Mr. Wayne Donald for affording me the opportunity to work in the I.T. Security Lab over the last several years and Dr. Patrick Schaumont for serving on my committee.

I would also like to sincerely thank Col. Timothy Buennemeyer for taking me on as a research colleague and letting me work on spinoff aspects of his dissertation work. His guidance and direction are greatly appreciated, as is his dedication and hard work that ensured the completion and success of the B-SIPS usability study.

Next, I would like to acknowledge and thank the OPNET Corporation for granting me a research license for their product.

Last, but certainly not least, I would like to thank my fiancé, my parents, and all of my family and friends who supported me through all the ups and downs I encountered over the last two years. I could not have accomplished this without those kind words of encouragement you were always armed with. Thanks!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

As technology progresses, mobile networking devices become more prevalent for both industrial and personal use. This increase in mobile device usage provides new venues for security exploits. Devices equipped with adequate computer processors and disk space, programs such as Norton Antivirus and MacAfee are available to protect and detect against malicious network traffic. Small mobile devices, such as PDAs and Smart Phones, however, lack the computing power, memory space, and operating system complexity to support such applications. The Battery-Sensing Intrusion Protection System (B-SIPS) research seeks to provide a solution such that these small mobile devices will no longer be defenseless to network attacks. This research extends B-SIPS by optimizing device battery lifetime and security, performing a usability study, and providing deployment suggestions based on data gathered from large-scale network simulations.

Prior to detailing the intricacies of B-SIPS or the optimizations performed upon that solution, an overview of network security terminology and capabilities is necessary. This introduction will provide the background necessary to describe the scope and applicability of this research. Section 1.1 will review basic network security concepts and goals. Section 1.2 then presents the basic B-SIPS concept and goals. Section 1.3 follows, providing an overview of the optimizations this thesis endeavor contributes to the B-SIPS goals. Finally, Section 1.4 lays out the thesis structure.

1.1 Network Security Terminology

When discussing possible solutions in securing wireless networks, the following terms are applicable: threat, risk, vulnerability, and attack. A threat refers to a possible security breach due to the existence of vulnerabilities, or security system flaws or weaknesses. The risk associated with the threat is the likelihood that attacks, or attempts to bypass security, launched on the associated vulnerabilities will be

successful [24]. Regardless of the resources poured into network security for a given device, application, or infrastructure, vulnerabilities and threats will always be present. While this is an undeniable truth, as no code or algorithm is invincible, minimizing the number and severity of these vulnerabilities can help to lower associated risks.

In order to minimize the vulnerabilities present, users and system administrators utilize *network security tools*. There is a vast array of such tools, each of which focuses on a specific aspect of securing the network. *Vulnerability scanners* send carefully crafted packets to test systems and devices for vulnerabilities. Example testing scenarios might include determining whether a network port, which is known to be associated with a particular network Trojan horse-style attack, is open, or whether a system running Windows XP has applied all of the available Windows Service Packs. Network administrators can determine whether company or campus policies are being followed by using the results found with these vulnerability scanners. If policies are not being followed and systems are left unpatched against these vulnerabilities, the systems may be broken into. In such cases, *Intrusion Detection Systems* (IDS) play the role of an intrusion alarm. In IDSs that are *host based*, the device monitors its own system level activities and notifies the system owner if any discrepancies are found. In IDSs that are *network based*, however, network traffic is monitored and compared to predefined attack signatures. If a packet or network connection is flagged as malicious, the network security administrator is notified via a management console tool [28]. *Intrusion Protection Systems* (IPS) are reactive IDSs, which alter their system settings to decrease the likelihood for similar attacks to occur in the future [32]. Frequently, IPSs protect against both predefined signature attacks and unknown, or anomaly, attacks. Like all tools, however, host based and network based intrusion detection/protection systems have pros and cons. Therefore, for maximum network security, this research initiative suggests the use of both host and network based intrusion protection systems (IPSs), with the idea that the two systems will compliment one another.

Many mobile device users do not comprehend the importance of network security on their PDAs. In network security, there are five basic areas that IPSs aim to protect: availability, integrity,

authentication, confidentiality, and non-repudiation [24]. These five areas are just as valid, and in need of protection, for small mobile devices as they are for servers and desktops.

Availability of a resource refers to device access, Internet access, and file / application access. Loss of availability to an Amazon server for a short allotment of time may mean a loss of thousands of purchases. In terms of mobile devices, loss of availability may mean that the device battery is drained and a real estate agent looses a bid on a home they are attempting to sell. Alternatively, an executive for a large firm may be planning to close a deal on a merger, but be unable to do so, due to a lack of availability.

Integrity of a system refers to the verification that data has not been modified. This is important on handheld devices, as tampering with data could cause the quantity of an order to be increased, a monetary transaction to be transferred incorrectly, or a downloaded image to contain a serious Trojan virus.

Authentication refers to identity verification. Users generally use Virtual Private Network (VPN) authentication prior to viewing any data sensitive to the company with which they are employed. Malicious network users may use mobile device backdoors to place keystroke loggers, or applications that will echo all characters typed by the user into a file or across the network, onto the device and capture the VPN password as the user authenticates. This password will then allow the hacker to gain access to the cooperate network and wreak havoc, all while masquerading as the original user.

Confidentiality indicates that only the intended participants are able to view a user's data and network transactions. A breach in confidentiality can lead to eavesdropping, man-in-the-middle attacks, and a loss of data integrity.

Finally, *non-repudiation* refers to accountability of a user. In essence, this is an endeavor to prove that certain data packets were sent and received properly, that a user did indeed purchase a specific product, or that a server authentication was indeed completed. A lack of non-repudiation, with respect to either

stationary devices or mobile devices, breeds a lack of accountability and leaves any lawsuits that arise from network security breaches without proof of origin [24].

The combined threats enumerated in this section require network security to exist. All security tools used by teenagers, employees, corporate managers, etc. have the aim to protect our devices such that data is available when we need it, has not been tampered with, is from whom we believe its from, is seen by only those whom we wish it to be seen, and is accounted for as it should be.

1.2 B-SIPS Concepts & Goals

Research described in this thesis builds upon and validates work done on Battery Sensing Intrusion Protection Systems (B-SIPS) [8]. B-SIPS is hinged upon that fact that smart mobile devices periodically obtain data pertaining to battery usage. A sudden increase in instantaneous current, without reasonable cause, can be an indicator that malicious activity is going on in the device. B-SIPS then warns the user, and transmits data to the B-SIPS server, which correlates this data with the data transmitted from other mobile devices in the area. Elaboration on all of these details will be provided in Chapter 2. The importance of the B-SIPS application is that it provides security for small mobile devices. These devices currently are unprotected and vulnerable to an increasing number of network attacks. This lack of protection is a side effect of the devices' inability to run standard IDSs, such as Norton Antivirus and MacAfee Antivirus, due to the simple operating systems, small memory capacities, and slow central processing units that make small mobile devices possible. Thus far, no applications have been deployed to remedy this gap in security. There have, however, been two endeavors to develop such a system. The first was the Battery-Based Intrusion Detection (B-BID) system devised by Jacoby [18], and the second is B-SIPS [8], which was built upon, and improved, the idea presented by B-BID. This research further builds upon the idea of securing small mobile devices; optimization intentions are outlined in the following subsection.

1.3 B-SIPS Optimizations

This research optimizes three areas of B-SIPS. The first optimization improves upon the B-SIPS client and server applications. In order to achieve application optimization, a usability study was constructed, performed, and analyzed. Pertinent results were passed on to the B-SIPS application development team, such that future iterations of the product will be improved. The second area of optimization is that of the data obtained from the smart battery. This part of the research investigates whether the detection capabilities of B-SIPS are increased when the data transmission rate between the device and the smart battery is varied. The smart battery polling rate investigation entails researching the Dell Axim hardware platform, devising MATLAB code to accurately model the Axim's platform, and finally running MATLAB simulations. Although the B-SIPS team could not implement the optimizations suggested because of limitations built into the hardware, the findings were published in conference papers, with the hope that hardware manufacturers may consider altering the polling rate of smart batteries in the future. Finally, the last optimization involved creating a large scale network simulation, and determining the optimum application transmission period for networks of varying sizes. Because throughput greedy applications are generally disabled in devices, the B-SIPS team wanted to ensure that the deployment of B-SIPS would not interfere with every day usage, which would then verify the likelihood of successful application deployments.

1.4 Organization of Thesis

The remainder of this thesis is organized as follows. Chapter 2 reviews background material and related work, while Chapter 3 further explores the optimization methodology behind this research endeavor. Chapter 4 then presents the B-SIPS optimizations performed on application usability. Next, Chapter 5 explores the optimization of smart battery polling rates, providing static and dynamic solutions to this problem. Chapter 6 explores B-SIPS application deployment by presenting large scale network simulations and showing how the B-SIPS application affects external network traffic. Finally, Chapter 7 draws conclusions regarding each of the B-SIPS optimizations, as well as the B-SIPS application package, its applicability in network security, and suggestions for future work.

2. RELATED WORK

Optimizing a Battery-Sensing Intrusion Protection System involves work in several distinct areas. For each of these areas, previous works have been accomplished. Section 2.1 will review previous works in the more general spectrum, which involves IDSs for small mobile devices; this section reiterates points made in Buennemeyer's dissertation and joint publications [8][7][3][5]. Section 2.2 will then investigate the advantages and disadvantages of present day network simulation tools.

2.1 Intrusion Detection Systems for Small Mobile Devices

Although security threats against mobile devices continue to rise, network security utilities for these power-constrained mobile hosts are not readily available, and are generally considered as an afterthought in comparison to service availability [9]. A complex problem for device designers is balancing the availability of the device, which is dependant on the usage time afforded by a single smart battery charge, and the security of the device being made available. As the security of the device is increased, the power consumption associated with this security is also increasing, thus decreasing device availability. One example of this can be seen by examining the possibility of establishing secure communication channels through authentication protocols. Although this action may have the advantage of increasing service accessibility and security, it also increases the device's computational and transmission requirements, leading to faster battery drain.

A technical specification for Advanced Power Management (APM) was developed to increase battery lifetime by managing device power for efficiency [21]. The APM application standard allows Basic Input Output System (BIOS) and operating system (OS) manufacturers to reduce energy consumption in their systems by including power management in both their BIOS and OS. The next stride in power management technology was accomplished by the Advanced Configuration and Power Interface (ACPI), which established industry-standards for power management interfaces being implemented in both OS and device hardware [1]. Next, members of the Smart Battery System Implementers Forum

devised the open systems communication standard, which specified the manner in which data sharing occurs between devices and their smart batteries [28]. Finally, the introduction of a Smart Battery Data (SBData) specification enabled devices to monitor their rechargeable batteries via the System Management Bus, a two-wire communication bus between the device and its smart battery [29][31].

In 1999 the concept of *sleep deprivation torture* through energy depletion attacks was suggested by Stajano and Anderson [30]. Such battery exhaustion and denial of sleep attacks comprise an emerging class of attacks that deprive users from accessing information on their devices by discharging the device battery without the users' knowledge [20][2]. Sleep deprivation and power exhaustion attacks accomplish this by rendering mobile devices incapable of shifting into appropriate reduced power states [7].

Martin et al. [20] categorized these battery attacks into classes: service-requesting, benign, and malignant. Service-requesting power attacks use the establishment of a large quantity of genuine service requests as a means to drain power from the responding device's battery. Benign power attacks attempt to activate power demanding processes to drain batteries rapidly. Finally, malignant power attacks alter device programs and applications to consume more power than necessary [7]. The prevalence of battery draining attacks will continue to increase proportionally to the deployment of mobile computers and devices. Documented battery draining attacks include the successful attack by Racic et al. [26], which "exploited vulnerabilities in an insecure multimedia messaging service, context retention in the packet data protocol, and the paging channel" without the knowledge of cellular phone users or network administrators [8][7].

Two defense systems were initially developed to mitigate such battery exhaustion attacks. The first of these solutions was developed by Nash et al. [22] and provided a battery constraints-based intrusion detection system (IDS) for laptop computer. This system "leveraged the laptop's robust computational power to estimate power consumption of the overall system based on metrics which included CPU load, disk read and write access, and network transmissions and receptions by using a multiple linear regression model" [8][7]. The solution then performed multiple linear regressions on the combination of these metrics and performance data from the operating system. These linear regressions afford the

system with correlation coefficients that enable power differentiation between various device components. Nash's system used this power differentiation to not only determine the power consumption of each process, but also as an indicator of possible intrusions. For maximum system benefit, this system needs to have properly configured thresholds.

The second system developed to mitigate battery exhaustion attacks was the host-centric Battery-Based Intrusion Detection (B-BID) system, which was developed by Jacoby [18] for mobile handheld devices. B-BID was comprised of the *Host Intrusion Detection Engine*, which detected battery behavior abnormalities, and the *Source Port Intrusion Engine*, which captured and correlated attack signature patterns. To our knowledge, B-BID provided the first feasible small mobile device IDS solution. The system was a success in terms of the ability to warn device users of possible intrusions, however, it lacked the ability to correlate data among multiple small mobile devices or report network trends to system administrators. Additionally, B-BID did not allow its system thresholds to be recalibrated by its users, which is important for systems being deployed on varying hardware platforms. The final downfall of B-BID is that it allowed users to configure the application for manual invocation of attack impedance. The likelihood of mobile device users constantly monitoring their devices, or having the reaction time to properly impede an attack fast enough to prevent substantial power depletion, is extremely low. Thus, B-BID does not provide an adequate solution for deployment, but does provide an excellent springboard for future development and research extensions.

B-SIPS research by Buennemeyer [8] developed an innovative battery power constraint-based model and system to help defend small mobile computers, smart cellular phones, and communication-enhanced Personal Digital Assistants (PDAs). B-SIPS extended concepts presented in B-BID by presenting a net-centric solution addition to the research. This extension provides not only the B-SIPS client application user with information pertaining to the attack, but also forwards this information to a CIDE server, where data from all mobile devices in the network can be correlated. Network Administrators are then able to use this correlated data to draw conclusions pertaining to the state of security in their network. Data correspondence between the B-SIPS client and server uses User Datagram Protocol (UDP), a best effort networking protocol that possesses low transmission

overhead, but does not provide the guaranteed delivery or verification of delivery that TCP does. The default transmission rate for information flow from the B-SIPS client to the CIDE server is 1 Hz, or one packet of information per second. This transmission rate is configurable via client application settings. In addition to the correlation of mobile device data via the B-SIPS net-centric CIDE server, B-SIPS provided research validations and application optimizations. To do so, a usability study was prepared and executed, showing that the B-SIPS applications were well received by participants and viewed as applicable and user friendly. Additionally, the study gave the development team feedback with which to improve future versions of B-SIPS. Optimizations in the smart battery polling rate showed that if hardware manufacturers altered their System Management Bus (SMBus) speeds in the future, the next generation of small mobile devices may have an even better chance of defending themselves against hackers. Finally, a large scale simulation showed that the B-SIPS application had little to no negative affect on the throughput of external applications on the network.

2.2 Network Simulators

The most significant portion of this thesis research involved validating and optimizing the deployment of B-SIPS. The development team for the B-SIPS client and CIDE server was able to perform testing on a deployment of 10 devices, but due to a lack of hardware, was not able to test the performance of the system on a large scale. This portion of B-SIPS optimization is meant to determine the feasibility of large deployments, as well as to suggest optimum data transmission rates between clients and server based on the deployment size and throughput requirements. The throughput of B-SIPS data is important in allowing the network administrator to adequately gauge malicious network activity, but just as important is the throughput of external applications; if B-SIPS is a hindrance or uses a high percentage of system bandwidth, users will disable the application. Therefore, if B-SIPS is to be deployed successfully in the future, deployment guidelines and suggestions will be an important aspect of its success.

Optimizations were made using scenarios created using an open source network simulator software called ns-2. Additionally, the scenarios created were designed to model the Virginia Tech network,

encompassing the appropriate access points, switches, and core routers that comprise the Virginia Tech wired-cum-wireless network infrastructure. Details pertaining to the simulation assumptions, construction, and results are discussed in detail in Chapter 6.

Large scale network simulations allow developers to make simplified assumptions regarding the impact produced by an application may when it is deployed on a specified network. Using such simulations provides statistics on theoretical networks that would otherwise be impossible to obtain due to a lack of network hardware. To date, there are two network simulation tools that are widely used in research. The first of these tools is ns-2. Ns-2 is "a discrete event simulator targeted at networking research" [12]. It is an open source Linux based simulation tool which uses tcl files to dictate the location and connection information between wired and/or wireless nodes. Various tutorials are available to demonstrate how to set up UDP and/or TCP connections, and the user forums are extremely active. To ns-2's advantage and disadvantage, many patches have been written to expand the current capabilities of an ns-2 version. This is an advantage because these patches allow simulation users to incorporate multiple wireless interfaces into their simulations. The disadvantage of these patches is brought about by a lack of backward compatibility. Maintainers of the ns-2 source code do not ensure the use of legacy patches in current releases of ns-2. This means that if there are two or more patches a simulation needs to use, there is a high probability that there is not a common version platform which supports both.

The second network simulation tool available is a commercial product called OPNET [25]. Graduate students may obtain a research license by submitting an application to the OPNET website. OPNET is a GUI based simulator, and it based on the concept of "click, drag and configure". Tutorials are available via the help menu option in the application, rather than endlessly browsing the Internet for tutorials, as is required by ns-2 users. Additionally, product documentation outlines how to use specific modules. My only critique on the OPNET tutorials and modules is that occasionally, the tutorial will show you how to perform a specific task, but will not specify what each of the configuration alterations you are performing actually does. To obtain this information a user must exert additional effort by searching through the documentation. Also, educational licenses do not include support, which means that aside from aiding graduate students with installation issues, there is nowhere to "go

for help" if issues or question arise. Discounted support rates are available, however, for many graduate students, funding for this support may not be available.

In summation, the simulation tools ns-2 and OPNET each come with advantages and disadvantages. Ns-2 lacks backward compatibility or high level application options. On the other hand, ns-2 has a large following in the academic world, and therefore active forums for advice. OPNET does not offer forums or advice to academic users, but does provide adequate tutorials, documentation, and FAQs. Additionally, OPNET is fairly intuitive, is run in a GUI mode, and even supports typical applications such as FTP and Voice Over IP (VOIP), as well as allowing users to create their own applications. Ultimately, OPNET proved to be the more appropriate of the two simulation tools to use.

As discussed previously, ns-2 was determined to not be adequate in representing a B-SIPS deployment in the Virginia Tech network since it is lacking required capabilities for such a simulation including multiple channel availability for wireless devices, infrastructure mode wireless communications, and user friendly packet tracing capabilities. Unfortunately, the ability to use multiple wireless channels (MW-Node patch) and the ability to use an infrastructure wireless environment (NOAH patch) require patches in contradicting ns-2 versions. Although I attempted to work around these issues in various ways, the end result was that it was infeasible to enable both patches as once. Additionally, packet tracing was provided by ns-2, however, there was no common format to trace packets traversing both wired and wireless interfaces. Thus, the ns-2 initiative was abandoned in favor of using OPNET.

3. APPLICATION USABILITY

This chapter describes the application usability study of B-SIPS, which has been deemed a viable measure of research scope and system success [8]. Product quality improvement in terms of feature functionality and intuitive graphical user interfaces served as additional motivation for performing a usability study. The execution of the stated usability study was a joint effort [8]. In this endeavor, Buennemeyer provided vision, motivation, and system knowledge, as well as residing as the lead participant interviewer [8]. Contributions included the shaping of documents to be reviewed by participants, the creation of the data collection tool, and result analysis. Section 3.1 provides the user with an introduction to application usability. Section 3.2 then discusses required study preparation. Finally, Section 3.3 presents the execution of the usability study and Section 3.4 reviews results and conclusions found.

3.1 Introduction to Application Usability

In practice, *usability* refers to the quality of user experience in terms of how easy a product or system is to use, and how useful it is. Specifically, usability includes the effectiveness, efficiency, learnability, safety, and user satisfaction of a product. Characterizing the usability of a system generally requires the indirect measurement of indicators such as task speed and correctness, user error rate, and user satisfaction [14]. A usability study is an iterative process that uses these indirect measurements to provide feedback on how the product may improve. Hartson refers to the Usability Engineer as a "priest in a parachute", meaning that the engineer is not meant to fix the issues with the system, or to oversee the future progress of the system, rather, the engineer is meant to "drop into [the] project, bless it, and leave quickly" [14]. In essence, the engineer should follow the guidelines given by the

developers or company to create appropriate testing methods, execute the usability study, and provide the results and appropriate fix suggestions.

The usability study presented in this chapter describes a first attempt at optimizing the B-SIPS client application and CIDE server. The primary restriction placed upon the study by the research team leader, Buennemeyer, was that only a single iteration of usability testing was possible. This restriction was put in place due to B-SIPS research time restrictions. Although the lack of multiple iterations contradicts the standard practice for usability studies, it does not disable the study from providing the necessary verification of validity and application usability flaw feedback, as the percentage of usability problems found is dependant on both the number of testing iterations and the number of test participants [23]. As Section 3.2.3 will show, the lack of testing iterations in the B-SIPS usability study is offset by the number of testing participants used. For an increased user-friendly and deployment ready version of the B-SIPS applications, however, subsequent iterations of this usability study may be required.

3.2 Preparing Usability Study Material

A successful usability study requires adequate preparation prior to any invitations for user participation. The first preparatory step is an in depth systems analysis [14]. This entails the usability expert familiarizing themselves with both the product being scrutinized, as well as any current tools the people at whom the product is being targeted at are currently using. Once a general understanding of the product and its targeted users has been accomplished, user groups must be defined and benchmark scenarios prepared; this process is outlined in Section 4.1.1. After benchmark scenarios have been defined, a data collection tool, such as the one introduced in Section 4.1.2., can be extremely helpful in accurately logging task duration time, as well as organizing collected user data. The next step in usability study preparation, which is discussed in Section 4.1.3., is to determine not only the quantity of participants desired, but also where the study will be advertised. Finally, all work discussed thus far must be presented to the Virginia Tech Institutional Review Board for approval (Section 4.1.4).

3.2.1 Creation of User Groups and Benchmark Tasks

This section introduces the concepts of user groups and benchmark tasks relative to B-SIPS. *User groups* can be defined as a class of people with similar training, knowledge, and application goals [14]. B-SIPS possesses two such user groups. The first is the general user, who can be characterized as a user with general computer knowledge, familiarity with mobile devices, an urge to protect their own devices and data, and a lack of professional experience as a network security administrator. The second user group is comprised of trained network security administrators who possess the need to protect all systems and devices in their subnet by quickly and correctly assessing vulnerabilities and hazards. Henceforth, the first user group will be referred to as *general user* and the second group as *system administrators*. In deployment scenarios, general users would primarily use the B-SIPS client on their mobile device, while system administrators are more likely to use the B-SIPS CIDE server.

The next step taken in the usability study was the creation of usability specifications, which includes both objective and subjective data. Objective date refers to observable user performance, and in respect to usability studies, is generally carried out via benchmark tasks, which will be introduced shortly. Subjective data, on the other hand, refers to user opinion and satisfaction, and generally comes in the form of user questionnaires or surveys [14].

The first documents created were the benchmark tasks and surveys associated with the B-SIPS client application. The B-SIPS usability study recorded all data obtained using the tool discussed in 3.2.2. These benchmarks evaluate the participant performing simple tasks, such as altering B-SIPS client configurations and starting its connection with the server, and progresses through more complex scenarios, such as determining system attributes once an attack has been detected. For each benchmark task, and benchmark sub-task, the time duration is monitored and recorded using the B-SIPS data collection tool. Following the completion of all client benchmark tasks, the user completes a survey. Results from the benchmark tasks and surveys will be reviewed in section 3.4.

Next, benchmark tasks and a survey were constructed for the B-SIPS CIDE server. The task complexities ranged from easy tasks such as determining the date and time of a recorded attack, to more complex tasks such as correlating data from the B-SIPS CIDE server with data collected from a secondary network security tool name Snort. The time to complete each benchmark task was recorded, along the response to any data prompts. Finally, participants testing the CIDE server were asked to complete a survey and make suggestions and comments pertaining to the B-SIPS system as a whole.

All benchmark tasks and surveys were constructed by Buennemeyer [8]. Each task was edited to ensure that participants are given adequate and sufficient information to complete the task, without trivializing the task by providing too much information. Additionally, the surveys were edited such that the participant would not be swayed by question tone or wording. Tasks were then performed by the B-SIPS research team to ensure their validity, and after final minor modifications, tasks and survey questions were programmed into the B-SIPS usability study data collection tool, which will be introduced in the following section.

3.2.2 Creation of Data Collection Tool

The B-SIPS usability study data collection tool was created as a means of reducing human error in terms of timing and transcription errors, easing the data collection process, and organizing participant and study results into Microsoft Word and Excel files. C# was chosen as the method with which to develop the tool, due to the ease with which it creates GUI interfaces. This section outlines the capabilities and features of the data collection tool.

In addition to creating a data collection process with a high ease of use, caution was taken to ensure that data is recorded accurately, future benchmarks do not influence the answers participants give for current benchmark questions, and the identity of each participant is protected once they leave their usability session. Additionally, caution was used during the creation of the data collection tool, such

that the benchmark questions used did not influence the user to answer in a specific manner, but recorded their understanding accurately. Specifically, the use of radio buttons, check boxes, and text boxes allow the data tool to specify to the user when to respond with a single answer, multiple answers, or when the question is intended to be open-ended. The use of these features cuts down incorrectly formatted, or unusable, data. Additionally, open ended questions enable the investigator the ability to fully test the user's knowledge by not allowing an opportunity to guess between provided answers, as is the case for multiple choice questions. Each benchmark task is given to the user separately to protect current answers from being influenced by later information given,. Usability studies done without such data collection tools accomplish this separation of questions by requiring each benchmark, or each portion of a benchmark, to be printed on separate sheets of paper. In such a scenario, no subsequent benchmark task papers are given to the user prior to the completion of their current assignment, and users are not allowed to revisit previous questions. Additionally, all desired benchmark timing must be accomplished using a stopwatch, which introduces a measure of human error. This measure of error, along with additional exertion required from the investigator to operate a timing device, is reduced in the B-SIPS usability study via the use of the B-SIPS data collection tool. The investigators are now free to utilize their time more efficiently, taking notes on the user's body language and verbal comments pertaining to the system. Additionally, no investigator prompting is required for a participant to continue onto a new benchmark task or subsection; the user simply presses the *continue* button located on the GUI. As the user proceeds through questions in their benchmark task, the previous question's answer field is grayed out and looses its capacity to be edited; a new question and answer field appear below it. The answer to the previous question, along with the duration spent on that question, is recorded to file. Prerecorded data cannot be connected to the any specific participant because each user selects a reference ID at the beginning of their session (Figure 1). This ID is the only link between the user and the data recorded during their session. The details pertaining to the data collection tool discussed thus far affirms the expectation that the tool ease the data collection process, while maintaining data accuracy and user anonymity.

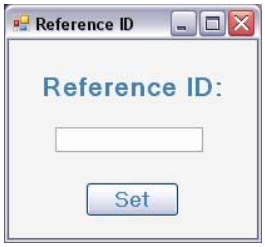


Figure 1. Usability study ID request

The next goal of the B-SIPS data collection tool is to simplify data organization. This data collection tool is compared and contrasted with the traditional pen and paper solution in order to highlight the advantages gained through its use. For traditionally collected usability data, benchmark task questions are answered on the sheet of paper that either depicts the scenario the user is being asked to perform or prompts the user to draw their own conclusions to the data seen before them. Any timing information collected by the investigator is noted down on a separate sheet of paper. Once the usability session is complete, the investigator files these papers away. After the entire study has been performed, the investigator may either sift between the papers to tally the data taken, or may enter all data into a spreadsheet for organizational reasons. The B-SIPS data collection tool, however, automatically generates Microsoft Word files and Excel sheets. The Microsoft Word file is created for each participant, such that each benchmark question is listed, followed by the answer given and the time required for that answer to be derived. This file looks much like the paper version a traditional investigator would file, only it is an electronic version, rather than a paper version. Additionally, an Excel spreadsheet is created that contains data from all users, where each row in the file denotes a new study participant. The first entry in each row denotes the reference ID discussed previously, while subsequent row entries denoted the question being asked, the answer given, and time spent on the question. In Figure 2 a small portion of the spreadsheet created is shown. The spreadsheet portion of the data collection tool makes it extremely easy to average numerical data and view general trends in both multiple choice and open ended questions. An additional Excel spreadsheet allows you to view in depth averages and statistics once the study is complete. This second excel sheet was not auto-generated, however, the effort required to create it was reduced greatly by the work that was automated. The organizations methods of the B-SIPS data collection enable usability study

investigators to conserve time and energy, thus fulfilling the expectation of simplified data organization.

Figure 2. Usability Study Excel Spreadsheet Data

Participant ID	Question	Answer	Question	Answer (Part 1)	Answer (Part 2)	Answer (Part 3)	Answer (Part 4)	Question	Answer
Q05.doc	2a	0	2b	NO	NO	NO	NO	2c	5 or less
R06.doc	2a	0	2b	NO	NO	NO	NO	2c	5 or less
J07.doc	2a	1	2b	NO	NO	NO	NO	2c	5 or less
S08.doc	2a	7 or more	2b	NO	NO	NO	NO	2c	10
Z09.doc	2a	7 or more	2b	NO	NO	NO	NO	2c	5 or less
N10.doc	2a	2	2b	NO	NO	NO	NO	2c	5 or less
T11.doc	2a	7 or more	2b	NO	NO	NO	NO	2c	5 or less
O12.doc	2a	7 or more	2b	NO	NO	YES	YES	2c	5 or less
I13.doc	2a	0	2b	NO	NO	NO	NO	2c	

Data integrity and organization is a vital aspect of any usability study. Without these attributes, data collected may be deemed either invalid or unusable. The B-SIPS data collection tool's compliance with these expectations made it extremely valuable to the study and was a worthwhile development endeavor.

3.2.3 Selection of Participants

Another vital aspect of a usability study is the selection of participants. It is important to have the proper number of participants, as well as the right kind of participants. This section discusses the determination behind each of these aspects and reviews the choices that were made regarding the B-SIPS usability study. Usability expert Dr. Jakob Nielson argues that while many studies suggest that testing as many participants as possible is ideal, a much smaller participant pool is necessary [23]. Nielson's research shows that each usability participant uncovers a proportion of the usability problems. This proportion is represented as L, and typically works out to be approximately 31%. As the number of participants, denoted as N, increases, so does the overlap in usability problems found. Due to this phenomenon, Nielson devised Equation 1 to characterize the probability that all usability

problems become exposed, based on the number of participants and the expected proportion of issues each participant will find [23]. Various values of N are shown in Figure 3.

Equation 1. Probability of 100% Usability Problems Exposure
$N(1-(1-L)^n)$

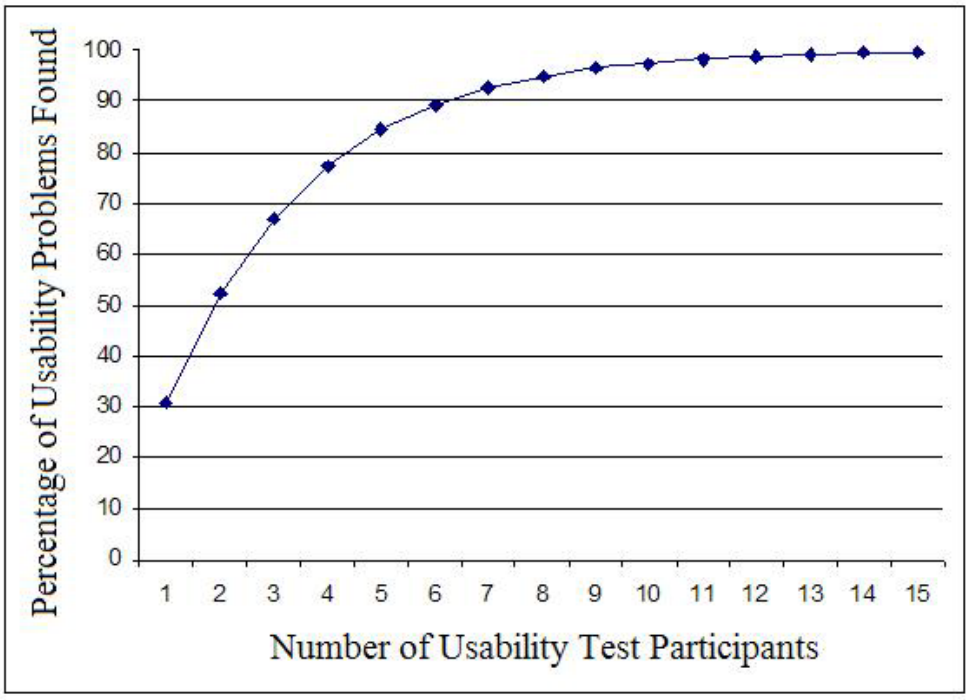


Figure 3. Determining the Proper Number of Usability Participants

Nielsen suggests that multiple iterations of usability testing are performed, where each iteration contains five participants. Thus, by finding approximately 85% of the issues with the product, fixing them, and then proceeding with another usability study, an optimal product may be devised [23]. Nielsen also mentions that in studies containing multiple user groups, additional participants must be obtained, such that there are an adequate number of qualified subjects for each user group. The B-SIPS suite of tools supports two user groups: general users and system administrators. Therefore,

participants identifying with each of these user groups would need to be obtained, in essence, doubling the x values in the graph shown in Figure 3. Nielsen's strategy dictates that each of three iterations of B-SIPS testing needs ten users, with five general user participants and five participants with system administrative backgrounds. As mentioned previously, however, there was a time constraint on the B-SIPS leadership and development team, which deemed Nielsen's method unfeasible. Due to this time constraint, the goal of the usability study was the determination of as many of the usability problems as possible, with a plan to fix a portion of them, and document the remaining issues for future generations of B-SIPS research students. This single iteration restriction dictates that the percentage of usability problems found would need to closely approach 100%, and will therefore increase the required number of participants for the study to be thirty, with half of the participants possessing a background in system administration.

The participant pool and study advertisement were determined next [8]. System Administrators known by members of the Virginia Tech IT Security Lab were contacted for participation in the study, as well as employees of 4Help, Virginia Tech's customer help center for students. Additionally, advertisements were spread via word of mouth to students participating in the Network Security course. Finally, graduate colleagues were contacted and asked to participate.

This section outlined the selection of usability participants. An overview of Nielsen's methodology provided approximate participant quantities necessary to obtain the desired exposure of usability problems. This information, in conjunction with previously defined iteration constraints, provided the usability team with the knowledge that thirty participants would be required. Participant publicity thrusts were then devised [8]. Two user groups were defined, the first being general users, which would be represented in the participant pool by Network Security students and graduate colleagues, and the second being system administrators, which would be represented in the participant pool by Virginia Tech system administrators and 4Help employees. Additional information regarding the backgrounds of the selected participants will be reviewed in Section 3.4.1.

3.2.4 Validation of Study via Virginia Tech IRB

One of the final preparatory steps for the B-SIPS usability study was obtaining approval from the Virginia Tech Institutional Review Board (IRB). This board assesses research studies in which human participation is required, ensuring that the research benefit outweighs all risks involved and that all participants are notified of any risks or benefits affecting them prior to their consent being given [16]. In the case of the B-SIPS usability study, an exempt review form was completed, outlining the goals of the study, the participants desired, the benchmark tasks which would be used, and a copy of the consent form (see Appendix A.6). The paperwork and request forms required for the Virginia Tech IRB were drawn up and submitted per IRB requirements [8]. Additionally, each of the B-SIPS team was required to complete the Virginia Tech Human Research Participant Protections Tutorial and quiz [16]. Once the research participant protections tutorial and quiz was completed and the IRB's approval of the study was received, the usability study was launched.

3.3 Usability Study Execution

The execution of the usability study refers to the process which took place between the entrance and exit of each participant. For each participant, this process began with a brief introduction to the B-SIPS research team and an overview of the product and product motivation. This product motivation included information regarding the lack of security on current cell phones and PDAs, along with the knowledge that network attacks specifically geared towards these devices have been devised and will become more prevalent in the future. Disclosure paperwork was then examined and signed. Next, the participant was provided with an overview on basic PDA usage, as well as manipulation of the B-SIPS application, after which the user was given access to a lab computer running the B-SIPS data collection tool. At this point, participants selected a reference ID, and completed an entrance interview that gathered information pertaining to user experience with PDAs, system administration, network security, job occupation, and level of education. Post background check, users were provided with a PDA which was running the B-SIPS client application, and were asked to complete various benchmark

tasks. These tasks, and their GUI representations, are shown in Appendix A. Once all client benchmark tasks were completed, sentiments pertaining to aspects of the application such as usefulness, ease of manipulation, and likelihood of regular use post deployment were obtained using a Likert scale questionnaire. Next, the participant was provided with an overview of the B-SIPS CIDE server. Several aspects of data manipulation and filtering options were reviewed, while others were left unexplained in order to determine user exploration and discovery. Once the server overview was complete, a new set of benchmark tasks, catered towards CIDE server use, was presented to the participant. A second questionnaire, catered towards server usability and manipulation, was presented to the user. Finally, the participant was provided with an open ended text field with which to elaborate on any relevant questions, concerns, or suggestions for future development of the system. Users are prompted for any additional feedback by the investigator, and are then thanked and excused.

Results and conclusions from the investigations, performed primarily by Buennemeyer and myself, with an emphasis on investigations by Buennemeyer, due to my relocation for summer employment are discussed in Section 3.4 [8].

3.4 Analysis and Conclusions

This section reviews and analyses the results compiled with the B-SIPS data collection tool, as well as providing insights as to the general usability of the products, areas of strength and weakness, and how to proceed with optimizing future versions of B-SIPS. The participant demographics will be outlined in Section 3.4.1. Section 3.4.2.discusses the data collected, issues found, and bugs fixed for the B-SIPS client application. Section 3.4.3 reviews data, issues, and bugs pertaining to the B-SIPS CIDE server. Finally, section 3.4.4 reviews the B-SIPS usability study as a whole and draws conclusions regarding the usability and relevance of the client application and CIDE server.

3.4.1 Participant Demographics

Usability study's findings are highly dependant on the experience and validity of the participants chosen. As stated previously, the B-SIPS participant pool was drawn from Virginia Tech system administrators, 4Help employees, Network Security students, and graduate colleagues. Confirmation that approximately half of the participants possessed the background to adequately review the CIDE server application, however, required knowledge pertaining to background specifics, such as user education and system administrator experience. As shown in Figure 4, roughly 30% of the participants possessed PhD degrees in computer related fields, while approximately 25% had earned or were pursuing bachelors or masters degrees in computer related fields. Additionally, Figure 4 shows that 45% of the participants had in excess of six years experience in system administration [5].

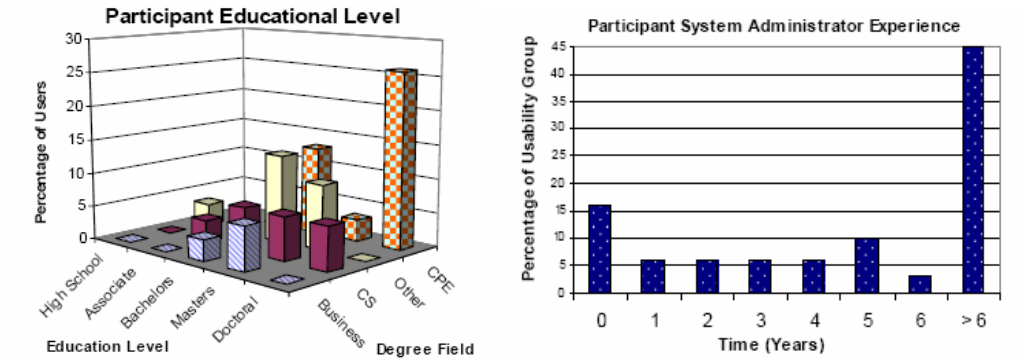


Figure 4. User Education and Experience

In regards to device knowledge, the majority of the participants expressed experience and comfort with using handheld devices, servers, and Intrusion Detection Servers, as shown in Figure 5 [5]. These statistics indicate that the desired user groups have been fulfilled and the results found in this study are relevant. Additionally, the relatively high percentage of non-computer related degrees helps to vary the average general user's knowledge and skill set.

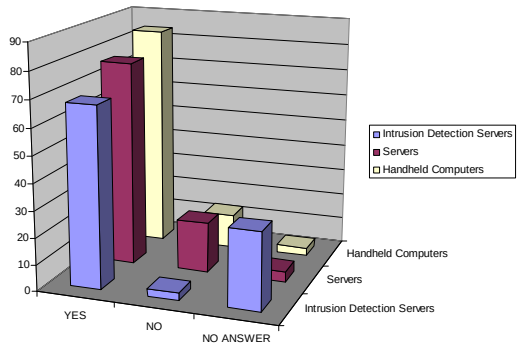


Figure 5. Participant Device Experience

The remainder of this section reviews results from the client and CIDE server application's benchmarks and surveys. Additionally, cost importance tables will be constructed for each of the applications and conclusions regarding the system as a whole will be drawn.

3.4.2 B-SIPS Client Application Results

Results from the B-SIPS client application indicate that it was received well. In general, users were able to manipulate the device as requested. In some cases, however, participants either took a long time to answer the question at hand, or additional aid was requested. Strong points included an understanding and interest in the need for such a service, and a positive attitude throughout and after their experience with the application. Problem areas stemmed from a lack of knowledge pertaining to network attacks, as well as a lack of confidence in their actions. These areas can be improved via additional user feedback, minor changes in labels, and hyperlinks from labels to the help file. Aside from these few setbacks, data from the client survey, shown in Figure 6, supports the B-SIPS client tool, giving it an average overall rating of 4 on a scale from 1 to 5 [5]. Additionally, this research suggests that participants are willing to not only use the application themselves, but advise others to use it as well.

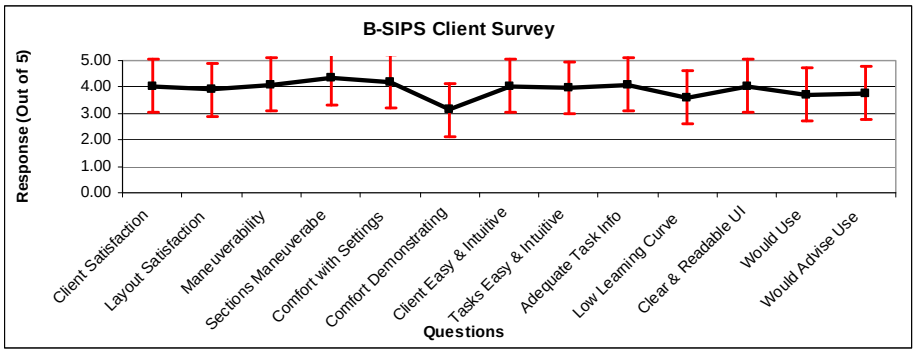


Figure 6. B-SIPS client survey responses

In order to greater increase the likelihood of success upon deployment, however, usability issues, such as those causing user lack of confidence, must be addressed. To organize and prioritize this data, a cost importance table (Table 1) was created following the guidelines taught in the Virginia Tech Usability Engineering course [14].

Table 1. B-SIPS client cost importance table

Client Problem	Importance	Solution	Cost	Priority Ratio	Priority Rank	Culmulative Cost	Resolution
Users very confused with the connections tab	M	Add popup "Bluetooth searching", eliminate ICMP button, and insert a label indicating that connection information appends to list	0.75	M	1	0.75	Fix
Users showed shock and lack of confidence in their actions during device calibration	M	Splash screen when calibrating and backlight restart after calibrating	0.75	M	2	1.50	Fix
No user notification of success when users press the set button	3	Add popup for "Set" buttons for done	0.25	1200	3	1.75	Fix
Users indicated that frequent pop-up alerts might get annoying	2	Turn off Alerts option in Settings	0.25	800	5	2.00	Fix, if time
Users were unsure of what many labels and buttons did, but did not think to utilize the help file	2	Hyperlink help file	0.25	800	6	2.25	Fix, if time
Users had a problem identifying attacks in the advanced tab	3	Adding "Attack" col to table (as the first column); color "attack" rows red	0.50	600	4	2.75	Fix, if time
Misspellings found	1	Fix typos	0.25	400	7	3.00	Fix, if time
Users can't match Microsoft process names to execution files	2	Add process display name to process list tab	1.00	200	8	4.00	Fix next version
Users annoyed at having to close all applications to review attack	2	Notification--old bat icon or on bottom bar	1.00	200	9	5.00	Fix next version
Keyboard disappears after recalibrate	1	Investigate & mitigate code issue	0.50	200	10	5.50	Fix next version

First, a list of usability issues, obtained via observing which tasks gave participants trouble, as well as from verbal comments and body language of the user, was compiled. Next, each problem was assigned an importance. In the cost importance table, an importance of 1 signifies a relatively unimportant problem and an importance of 3 signifies an important problem. Usability issues in which addressing and fixing the problem is mandatory were assigned an importance of 'M'. Once the assignment of importance was complete, I conferred with the B-SIPS development team and determined solutions for each problem.

As an example, the second item in the B-SIPS client's cost importance table will be reviewed. Benchmark task 2 prompts the user to recalibrate the system, and follows the question up by asking the participant to explain what effect their action had on the system. Though most of the users were able to complete this task fairly easily, the majority of the participants showed signs of confusion when the device's screen backlight faded. A reasonable solution to this problem is to flash a splash screen informing the user that the system is recalibrating and that they will be informed once the recalibration is complete. This alteration can be implemented quickly and easily, and is therefore assigned a cost of 45 minutes, or 0.75 manpower hours. Likewise, other client problems are assigned costs which reflect the effort and time required to fix them.

Next, a priority ratio is calculated for each problem. To accomplish this, the importance is divided by the cost and multiplied by 100; for mandatory usability issues this step is skipped, due to an inability to multiply M with a fraction and obtain a real number, and the priority ratio is set to M, or mandatory. The client problems are rearranged such that mandatory problems are at the top of the table and the remaining issues are listed in descending order, according to their priority ratio. Next, the priority rank numbers the client problems in descending order, and the cumulative cost column tracks the total number of manpower hours required as additional problems are fixed. Finally, resolutions are assigned to each usability issue. Due to limited client utility development time, the manpower time allotted to fixing bugs in the present version was set to three hours. The first three issues were deemed to be the most important, due to their high priority ratios, and were therefore deemed fixable. Once these bugs

are fixed, the development is encouraged to proceed in fixing bugs marked "fix, if time" with their remaining manpower hours. Finally, all issues listed under the dividing line were considered to be next version revisions.

Revisiting the issues listed in the client application's cost importance table, one will notice that several problems dealing with a lack of user feedback were addressed. Previously, the connections tab had confused the participants, whom had received no confirmation that their connection had been successful. As discussed earlier, recalibration held a similar lack of user feedback, as did the application of a "set" button when users altered system settings. Each of these issues has been addressed and corrected, as shown in the 'Solution' column of Table 1, due to the insight brought to the B-SIPS group from the usability study. Additionally, several typos were caught, hyperlinks were added to all application labels, an option for turning off alerts was added, and attack information was color coded to stand out more to users of the product. Overall, the study was able to not only highlight the positive opinion of users, but also help the development team improve the application and address the flaws that it possessed.

3.4.3 B-SIPS CIDE Server Results

Similar to the B-SIPS client application, the CIDE server was well received by the usability study participants. Again, most benchmark tasks were obtainable for participants, though there were several areas of frustration, such as poor labels, an extremely sparse help file, and a lack of sorting mechanisms. Overall, however, the participants rated their usability experience and satisfaction rate very well, and users noted that the standard Windows interface afforded them a comfortable environment in which to work. Server layout and workflow were also rated highly in the survey, which presented in Figure 7 [5]. Additionally, participants commented that the CIDE server enhanced security capabilities by providing a hybrid IDS solution for monitoring devices and would be increasingly relevant to future mobile device owners [5].

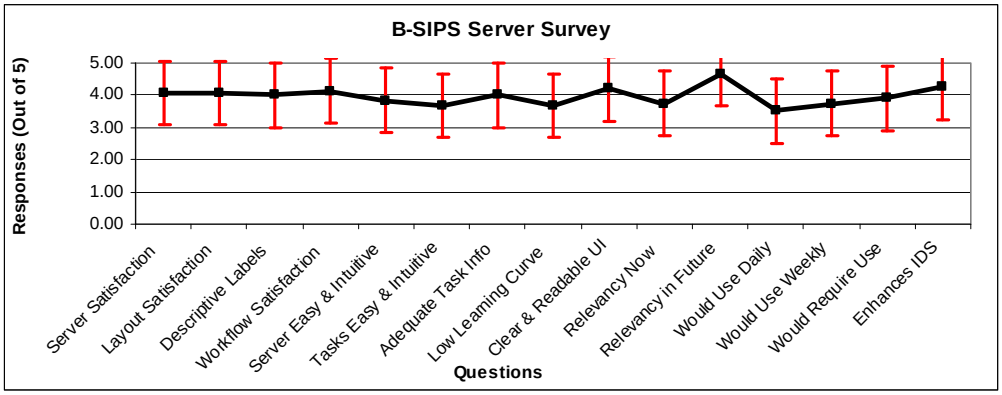


Figure 7 B-SIPS server survey responses

The usability study also uncovered a great deal of information as to how the server application could be improved. Data collected was used in the construction of a second cost importance table. This table lists the problems, suggested solutions, and importance relevant to the CIDE server, and may be seen in Table 2. The main issues with the CIDE server included poor labeling and instructions, as well as a lack of functionality in places where participants assumed it would be present. Specifically, users were disappointed in the lack of detail in the server's help file, and were surprised that the server provided no filtering mechanism. Additionally, several of the capabilities that the server application did encompass presented themselves to be challenging as the user interface for the capabilities was not clear or intuitive. Once solutions to these issues were devised by the usability expert and developer, a reasonable manpower hour value was determined to be approximately five hours. All usability problems below the five hour marker in the table were considered to be irritations, rather than detrimental defects, and were written off to be addressed in subsequent versions of the application. The remaining bugs were addressed and fixed, as specified in the 'Solution' column of Table 2.

Table 2. CIDE server cost importance table

Server Problem	Importance	Solution	Cost	Priority Ratio	Priority Rank	Culmulative Cost	Resolution
Help file contained no helpful info	M	Add help file	1.00	M	1	1.00	Fix
Users surprised that they could not sort data	M	Add sorting capabilities to database & live views	1.00	M	2	2.00	Fix
Many users did not understand the meaning of the tolerance feature	M	Explain tolerance feature	0.25	M	3	2.25	Fix
Users did not intuitively know that they could right click to sort analysis	M	Label in Analysis tab indicating right-click offers data sorting options	0.25	M	4	2.50	Fix
No way to know where data in the left pannel of the Correlation tab was coming from	2	Label pannel side as B-SIPS	0.25	800	5	2.75	Fix
Client time not dependable; server time not given	2	Add server time stamp to logs	0.25	800	6	3.00	Fix
User concerned about ease of report creation due to fields not being copy / paste enabled	2	Allow all fields to be copy / paste	0.50	400	7	3.50	Fix, if time
Time was given from Live Data, but no date	1	Live Data -> Add "Date" column	0.25	400	8	3.75	Fix, if time
Correlation lacks a 'count' column or a way to easily determine information about a selected chunk of data	1	Add attack and time counters, as well as displaying the time range and number of rows selected	0.50	200	9	4.25	Fix, if time
Users confused that they cannot select / view multiple correlations	3	Make it possible to select multiple correlations	2.00	150	10	6.25	Fix next version
Clicking on a graph or data table doesn't highlight information in the corresponding graph / table	3	Add brushing/linking	3.00	100	11	9.25	Fix next version
Graphs are confusing, need to be enlarged, and should give more info via user interaction	1	increase test size, add "time" to gumball info, and enhance mouse-over tool tips	1.75	57	12	11.00	Fix next version
Users annoyed with tolerance changes not being retroactive	3	Make tolerance retroactive	7.00	43	13	18.00	Fix next version

At a glance, the B-SIPS server had more usability issues to be addressed than the B-SIPS client did; that statement, however, is neither a negative nor a surprising fact. The CIDE server is much more complex than the B-SIPS application, and holds a great deal of information. Prior to a user session, several of the complaints the participants had would be hard to anticipate. For example, it was brought to the team's attention that all fields in the server would serve a system administrator better if they were copy / paste enabled. This feature allows the application's user to copy portions of the data into forms for documentation purposes. The request for this feature to be added was extremely valid, as were other requests and complaints, but less likely to be anticipated than some of the problems from the simpler user interface on the mobile device. That being said, the number of issues found on the

CIDE server was positive, in that the development team was given a great deal of insight to improve the application. Additionally, the defects found and feature addition requests did not prohibit the participants from completing their tasks or degrade their opinion of the system. Therefore, the CIDE server portion of the usability study was successful.

3.4.4 Usability Study Conclusions

The B-SIPS usability study analyzed the perceived validity and usability of B-SIPS research endeavor. To accomplish this, an appropriately educated participant pool was recruited, accurate and organized data measuring techniques were used, and evaluations were methodical and realistic. Sections 3.2.1 and 3.2.3 show the user groups and participants were carefully selected to be an adequate representation of those who would use B-SIPS in a deployment scenario. Educated students and employees will be likely to download and use the B-SIPS client application in an attempt to safeguard the battery lifetime of their device, data being stored on the device, and sensitive data which may be transmitted from the device to web applications or company servers. Computer repair employees and system administrators are likely to be the individuals interested in utilizing the CIDE server application, and are also the correct people to point out any flaws or features which would be deemed necessary for deployment. Once this desired participant pool was compiled and the Virginia Teach IRB approved the study, the developed data collection tool was used to compile all benchmark data, as well as suggestions, comments, and concerns. Finally, the data was reviewed by the usability expert and the development team to determine the appropriate solutions and priorities of each usability problem found. Reasonable goals were set, and less important tasks were placed on hold for future versions of each of the applications. Major usability issues were addressed and corrected and the B-SIPS client and CIDE server applications received validation of worth by a wide span of users [8]. Additionally, this work was published in the 41st Annual Hawaii's International Conference on System Sciences [5].

4. SMART BATTERY POLLING RATE

The smart battery polling rate refers to the rate at which a small mobile device utilizing a smart battery obtains information from its battery. The optimization of this rate, in conjunction with the use of the B-SIPS applications, provide the device with increased network security capabilities and device charge lifetime. This optimization was proposed by Buennemeyer and is explored here as a supplement to the B-SIPS research endeavor [8]. Goals pertaining to this area of research are not relevant for immediate implementation, as hardware restrictions in current implementations of the smart battery disallow implementation, but will instead be relevant for future models of smart batteries, should the proper battery design modifications be adopted.

Devices utilizing smart batteries receive periodic information updates from their batteries pertaining to attributes such as battery lifetime, instantaneous current and average voltage. B-SIPS utilizes this data by monitoring the spikes in the sampled instantaneous current that cannot be associated with a user approved process. Such spikes can indicate malicious network traffic, which keeps mobile devices in a state of high activity for extended periods of time, depleting battery power faster than normal and decreasing the battery lifetime [7]. Currently, smart batteries transmit device attributes once per second, or slower. Were they to transmit this information more rapidly, however, the B-SIPS client application would gain the capability to detect, and therefore disarm, malicious network traffic and attacks in a more timely fashion. This early detection would, in turn, decrease the malicious power drain associated with the attacks and increase the *battery lifetime*, which we define as the duration for which a fully charged battery is useable [7].

Simulation models were constructed using MATLAB to explore this optimization opportunity. These simulations operated based on two assumptions. The first of these assumptions specified that the Dell Axim X51 as the hardware platform being modeled. This Axim was selected because it was the PDA model utilized by students at Virginia Tech enrolled in the Wireless undergraduate course, and was therefore readily available for manipulation and testing by the B-SIPS research team. Additionally, for purposes of simulation simplifications, once a spike indicating malicious traffic has been detected, the

attack is considered to be disarmed. Considering that any incoming network packet can be the one which triggers the current sampling spike, the number of malicious network attacks a device experiences during its battery lifetime is termed its *attack density*. Thus, a simulated device with an attack density of n and a smart battery polling rate of r, where the rate is measured in polls per second, will have heightened periods of battery depletion for a total time allotment of $n \times r$. As stated, current implementations of smart battery equipped mobile devices typically poll the battery once a second, resulting in heightened power usage, due to network attacks, for n seconds. Although hardware restrictions disallow the B-SIPS client from altering the rate r at the current time, this research explores how future alterations to the smart battery might further enhance the capabilities of B-SIPS.

Section 4.1 provides a more in depth introduction to smart battery data collection and the methodology used in optimizing smart battery polling rates. Section 4.2 explores static solutions for simulated networks, while Section 4.3 shows how the polling rates produced from the previous section vary when applied to real networks. Next, Section 4.4 explores optimizations through the use of a dynamic polling rate solution. Finally, Section 4.5 discusses and draws conclusions regarding the three types of smart battery polling solutions presented in this chapter.

4.1 Introduction to Smart Battery Data Communication

Prior to creating models for simulating smart battery polling rate algorithms that will increase the security and battery lifetime of a device, hardware standards, specifics, and limitations pertaining to how the battery data is transferred must be explored. As mentioned previously, this study focuses on the hardware specifics pertinent to the Dell Axim X51, though the constants placed in the simulations provided can easily be replaced with values consistent with other platforms and the simulations rerun. This section focuses on flushing out the constants which will be required in order to accurately simulate a network using the selected mobile devices.

The first order of importance is to determine the maximum polling rate, in terms of seconds between smart battery polls. To do so, the data transmission rate and number of data bits being transferred

between the smart battery and the device using it must be determined. Present day smart batteries use the SMBus to transfer data [31] to their associated devices at a transmission rate of 100Kbps [10]. The data transferred consists of not only the instantaneous current that B-SIPS uses in its attack probability calculations, but also data pertaining to battery voltages, the percent of battery life remaining, battery temperature, etc. All data transferred, along with the variable value size, is shown in Table 3. Additionally, Thompson explains that each byte transmitted from the smart battery grows to a size of 39 bits once headers have been added, while transmitted words increase in size to 48 bits [31]. Applying these values into our calculations, there are 879 bits of data transferred across the SMBus at a rate of 100Kbps, yielding a transmission time of 8.79ms. Thus, if the smart battery were devised in such a manner that there was a constant flow of data from the battery to the device, the smart battery polling rate would be once every 8.79 ms (114 Hz). This is the maximum polling rate.

Table 3. Smart Battery Data

Power Status Calls	Data Size
ACLineStatus	Tx_Byte
BatteryFlag	Tx_Byte
BatteryLifePercent	Tx_Byte
Reserved1	Tx_Byte
BatteryLifeTime	Tx_Word
BatteryFullLifeTime	Tx_Word
Reserved2	Tx_Byte
BackupBatteryFlag	Tx_Byte
BackupBatteryLifePercent	Tx_Byte
Reserved3	Tx_Byte
BackupBatteryLifeTime	Tx_Word
BackupBatteryFullLifeTime	Tx_Word
BatteryVoltage	Tx_Word
BatteryCurrent	Tx_Word
BatteryAverageCurrent	Tx_Word
BatteryAverageInterval	Tx_Word
BatterymAHourConsumed	Tx_Word
BatteryTemperature	Tx_Word
BackupBatteryVoltage	Tx_Word
BatteryChemistry	Tx_Byte

Next, device constants for the mobile device were determined. Table 4 displays constants, as specified by the Intel PXA270 Processor data sheet [17]. The device voltage is important for unit conversions, while the battery lifetime gives us the power quantity to use for a fully charged device. Finally, the active and idle processor values differentiate the power used by the device when the device is and is not being attacked by malicious network activity.

Table 4. Dell Axim X51 Device Constants

Device Voltage	3 V
Battery Lifetime	1100 mA
Idle Processor	0.07222 mW
Active Processor	0.25694 mW

This section determined constant variable values to be used in the optimization of smart battery polling rates, as well as the maximum polling rate for the smart battery used in Dell Axim X51. The constant variable value will be used in simulations as the fastest possible polling rate. A polling rate where the battery is polled once every ten minutes, 0.001667 Hz, will be used as the slowest possible polling rate. Additionally, device constants were determined via the processor data sheet corresponding to the Dell Axim X51. Finally, it was determined that future implementations of the smart battery, geared towards aiding security via Battery-Sensing Intrusion Protection Systems, could mitigate timed attack by randomizing when in the polling interval the instantaneous current sample is taken. With these statistics collected and issues addressed, simulation models were ready for creation.

4.2 Static Polling Rate Optimization for Simulated Networks

In this section we introduce a MATLAB simulation model that will characterize polling rate optimizations in varying networks. This first model is very simplistic, however, as the sections of this chapter progress, the models will become more complex. Specifically, this section of chapter 5 explores how the alteration of the smart battery polling rate can improve the lifetime of the device when the network attack density is predetermined.

In general, there are two parameters that should be taken into consideration when constructing a testing platform for smart battery polling schemes [7]. The first of these parameters is the polling rate, which can be measured as either the time between battery polls (sec) or the in polls per second (Hz). Modifying the polling rate allows the smart battery device the opportunity to improve one of two

mutually exclusive power saving techniques. The first power saving technique is accomplished by polling the smart battery only as often as necessary, thus reducing the power overhead involved in performing the data transfer. The second parameter that should be taken into consideration is the *attack density* of a network, which quantifies the volume of network attacks experienced by a mobile device over a single smart battery discharge periods; in the case of Dell Axim X51s, this is equivalent to the number of attacks the B-SIPS device identifies as threshold violations between the time where the smart battery holds a charge of 1100 mA and the time when the battery is depleted. Devices in *low attack density* networks benefit from the first power saving technique, *overhead reduction,* in their smart battery polling interval. The second power saving technique, *rapid attack detection*, provides protection to devices that reside in networks with a large percentage of malicious traffic [7]. In such *high attack densities*, devices reduce the time their processors remain in the active power state by detecting malicious traffic as quickly as possible. This extent of this advantage is possible due to the assumption that attacks are disabled immediately upon detection; as the time to disable a particular network attack is indeterminable, particularly if the device user is not present, this assumption allows the simulation to flatten the attack playing field and also allows the mobile devices being simulated an effective way to capture and disable malicious traffic in a timely fashion [7].

The mutual exclusion of overhead reduction and rapid attack detection is dependant on the state of the network attack density. Rapid attack detection polls the smart battery frequently, thus allowing the device to catch malicious traffic quickly and efficiently. The drawback to this scheme, however, is the overhead involved in this rapid polling of the smart battery. The converse is also true. Overhead reduction reduces the power used in transmitting data from the smart battery to the device, but, due to its slow polling rate, is not effective in catching malicious traffic in a timely fashion. A compromise must be made. Calculating the more efficient use of the smart battery, while maximizing the security provided by B-SIPS, therefore requires a simulation that varies both the battery's polling rate and the network attack density [7].

```
for (1/10 sec; 1/10 sec; 600 sec)
    battery_attack_time(sec) = polling_rate(sec) * num_attacks
    battery_attack_mW = battery_attack_time(sec) *
        processor_active_rate(mW/sec)
    battery_attack_mA = battery_attack_mW / vcc_battery
    battery_remaining_mA = total_battery_lifetime(mA) -
        battery_attack_mA
    battery_remaining_mW = battery_remaining_mA * vcc_battery
    polls_per_min = 60 (sec/min) / polling_rate(sec)
    time_spent_polling = polls_per_min * battery_poll_time
    battery_mW_per_min = (time_spent_polling * processor_active_rate)
        + (time_not_spent_polling * processor_idle_rate)
    battery_remaining_mins = battery_remaining_mW /
        battery_mW_per_min
    lifetime_mins = ( battery_attack_time(sec) / 60 (sec/hr)) +
        battery_remaining_mins
    if lifetime_mins > max_lifetime
        max_lifetime = lifetime_mins
```

Figure 8. Static optimum polling rate determination pseudo code

Due to its simplistic development interface and its capability to graph results easily, MATLAB was chosen to program the smart battery polling rate optimization models. A simulation based on the pseudo code, shown in Figure 8, was developed to calculate the optimum polling rate for the network attack densities: 0, 1, 10, 100, 1000, 10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000, and 100000 [8]. For each network attack density specified, a simulation 60000 using polling rates was run, where the polling rates ranged from once every 10 minutes (0.00167 Hz) to once every 0.01 seconds (100 Hz). Once the simulation was complete, the battery lifetime calculations from all network densities and all polling rates were graphed. Results are presented in Table 5 and Figure 9 [7]. The table displays the value of the optimum polling rate for each network density in both seconds and hertz. The graphical representation denotes these optimum polling rates with a black circle [7].

These simulation results are limited in their applicability, however, due to the face that in real networks, network attack densities will not be predetermined. This information, however, will be useful in the real network simulations presented in Section 4.3.

Table 5. Network Density Optimum Polling Rates

Number of Attacks	Optimum Polling Rate		Number of Attacks	Optimum Polling Rate	
	Sec	Hz		Sec	Hz
0	600.00	0.0017	40,000	0.08	12.5000
1	20.02	0.0500	50,000	0.07	14.2857
10	6.32	0.1582	60,000	0.06	16.6667
100	1.98	0.5051	70,000	0.06	16.6667
1,000	0.61	1.6393	80,000	0.05	20.0000
10,000	0.18	5.5556	90,000	0.05	20.0000
20,000	0.12	8.3333	100,000	0.04	25.0000
30,000	0.10	10.0000	*	*	*

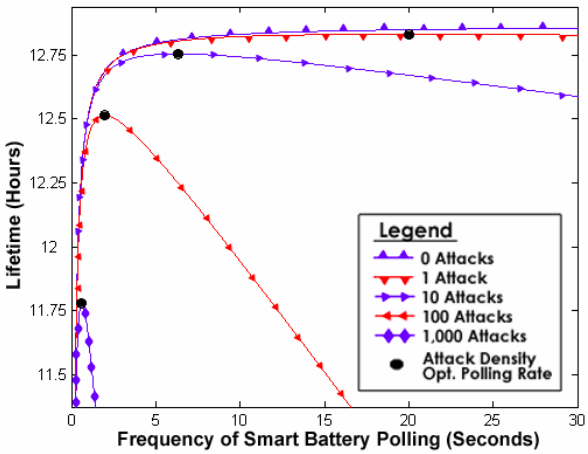


Figure 9. Effect of Battery Polling on PDA Lifetime

4.3 Static Polling Rate Optimization for Real Networks

The battery lifetimes of devices in real networks were determined using the calculation results from the last section. Although network densities cannot be predetermined, the rates determined to be optimal for simulated networks can be tested in real network conditions. A second MATLAB model was created, in which the optimal polling rates from Section 4.2 were tested for their ability to handle all 15 values of network density, rather than solely the network density they were desired to cater to. Code for this simulation is shown as Figure 28 in the Appendix. Running this simulation model sequentially, with each polling rate determined previously, affords the user with an accurate graphical representation of the discharge time of a device under varying network attacks. This graph is shown in Figure 10 [7]. Note that the x coordinate denotes the number of system attacks, and is shown as an exponential product of 10^4. Code pertaining to these simulations can be viewed in Appendix B, Figure 29.

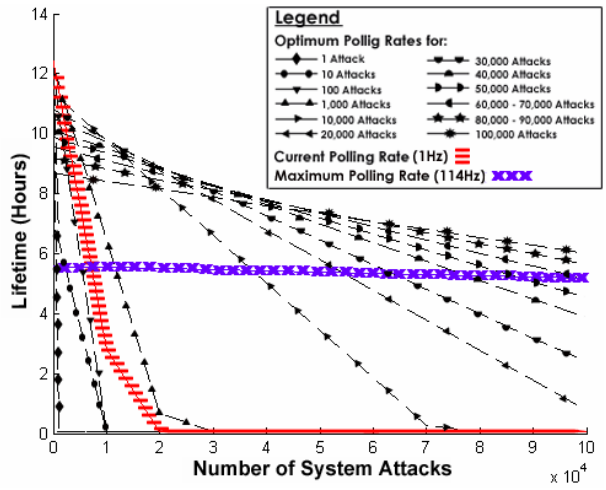


Figure 10. Device Lifetime Per Attack Density

The bold hashed line, shown in red, represents the Dell Axim X51's currently implemented smart battery polling rate, which is equal to 1 Hz. When the network's attack density is greater than 15,000 this polling rate scheme depletes the smart battery in less than an hour. Speedy battery depletion of this manner is referred to as *battery lifetime grounding*. On the opposite end of the spectrum, the bold cross-

hashed line, shown in blue, represents the battery lifetime associated with the maximum battery polling rate, as determined in Section 4.1 (114 Hz) [7]. In this scheme, the network attack density barely affects the battery lifetime. This scheme is unreasonable because the battery lifetime does not exceed six hours under any circumstances. As a compromise between the two extreme polling schemes, the black dashed lines represent the performance of the optimum polling rates under various network attack densities. Polling rates optimized for network attack densities greater than 10,000 provide the best compromise, offering lifetimes slightly shorter than the current polling rate under low attack densities and drastically increasing the number of network attacks required to ground the battery lifetime [7].

The static polling rates simulated in this section reinforce our earlier hypothesis that overhead reduction and rapid attack detection are mutually exclusive energy conserving methods [7]. Although several of the optimum polling rates found in Section 4.2 performed well under all simulated network attack densities, the results found in this section led to a new hypothesis. This hypothesis was that a dynamic polling rate, that has the capability to alter itself according to current network conditions, would most likely produce the best solution for smart battery polling. Dynamic solutions will be explored in Section 4.4.

4.4 Dynamic Polling Rate Optimization for Real Networks

This section discusses the prospect of increasing security and smart battery lifetimes via dynamic battery polling rate solutions. Theoretical advantages and disadvantages of dynamic solutions are examined. The primary advantage of dynamic solutions over static solutions is that they are able to adapt in varying network conditions. While the static solutions devised in the previous section can be optimized for specific network attack densities and then applied to networks with varying network densities, dynamic solutions have the ability to learn from current network activity, adjusting the current polling rate as necessary. This capability, correctly devised and implemented, should be able to optimize device lifetimes across a vast array of network conditions [6].

The drawback associated with this increase in productivity and energy efficiency is that the bus connecting the mobile device to its smart battery will require increased hardware complexity. In order to utilize the static smart battery polling rates introduced in Section 4.3, future smart batteries will need to alter their hardware to accommodate the new polling rate schemes presented in this chapter. When related to static solutions, this simply requires the hard coding of a new polling rate value for transmissions between the device and smart battery. For the dynamic solutions presented in this section, however, the hardware alterations are more substantial and complex. In this case, the hardware modifications must support bidirectional communication on the SMBus, such that the smart battery can read a rate value sent by the B-SIPS application and adjust its rate accordingly. Additionally, schemes would need to be devised and implemented to test the affects of the new hardware configuration, ensuring that the averages calculated by the smart battery, such as average current and temperature, are not distorted. Such distortion may possibly occur if calculation divisors are statically based upon the current polling rate. Once these hardware complexities are addressed, tested, and resolved, the exploration and implementation of the dynamic polling rate presented in this chapter can begin.

The TCP slow start windowing algorithm was used as the motivation for the dynamic polling rate solution suggested for use with the B-SIPS. The slow start windowing algorithm reduces the current window size to a pre-determined minimum value when contention is detected. Once the reduction occurs, the window size is doubled until it reaches a threshold, after which it is linearly incremented [19]. Similarly, the dynamic polling algorithm presented in this section defines two threshold polling rates. These thresholds were taken from static polling data. The minimum threshold defines the minimum time between attribute readings and is set to the optimal polling rate for devices experiencing network densities of 100,000, which was found in the previous section to be 25 Hz. The maximum threshold used in the B-SIPS algorithm refers to the maximum time between smart battery polls and is set to 0.05 Hz, which was determined to be the optimal polling rate for network densities of 1. The B-SIPS detection of an attack causes the current polling rate to be reset to the minimum polling rate threshold. The lack of attack detection causes the current polling rate to be doubled, with a ceiling polling rate of the maximum threshold [6].

Dynamic polling lifetime simulations were more complex than their static polling rate counterparts. While the lifetimes associated with attacks that are detected by static polling solutions are independent of attack timing, this attack timing is extremely important in determining the severity of the network attack in dynamic solutions. The extremes in attack timing can be characterized as falling in between two extreme cases. To model each of these extremes, equations for *clustered attacks* and *distributed attacks* were developed [6]. Clustered attacks provide a best-case scenario for dynamic solutions. Attacks are transmitted in a non-interrupted stream, minimizing the current polling rate during the device attack period and allowing the polling rate to ramp up to the maximum threshold afterwards, affording the most power conservation. Alternatively, distributed attacks are the worst case scenario for devices employing dynamic polling rate solutions. In this type of timing attack, the current polling rate is reduced to the minimum threshold via the B-SIPS' detection of an attack. The next attack is delayed just long enough for the mobile device to have returned to the maximum polling rate. Spacing the attacks in this manner allows the attacks to wreak the most havoc, as it keeps the device from remaining at the maximum polling rate for long periods of time. Logically, real world network attacks will fall between these extremes [6].

A *MATLAB* analytical model, which can be seen in Appendix B, Figure 29 calculated the lifetimes associated with the dynamic algorithm developed here. This model determined the lifetimes associated with both clustered and distributed attacks, using the same network attack densities as presented in previous sections of this chapter. In order to compare static and dynamic solutions, results were plotted together, as shown in Figure 11 [6].

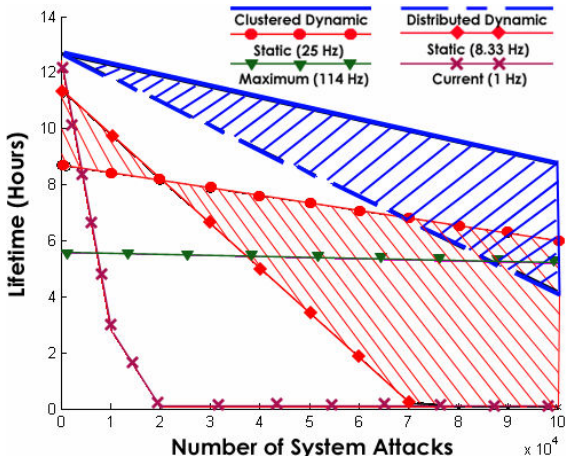


Figure 11. Effect of polling approaches on smart battery lifetime.

This figure again shows the rapid degradation of lifetime when the currently implemented smart battery polling rate undergoes high network attack densities, as well as the unacceptably low but steady lifetime associated with the maximum polling rate. Next, the set of most reasonable optimized static solutions are shown. The polling rates deemed most reasonable fell between 8.33 Hz and 25 Hz. These polling rates improve the lifetime of the mobile device significantly, with the marginal drawback of reduced lifetimes for networks with low network attack densities. Finally, dynamic solutions for both clustered and distributed attacks are shown. The lifetimes associated with the dynamic solution are exceptional and greatly outperform static solutions.

This subsection explored the theoretical advantages and disadvantages of dynamic solutions to smart battery polling rate alterations, presented the two attack timing scenarios characterizing dynamic lifetime calculations, devised and executed a *MATLAB* simulation model, and finally, compared the graphical lifetime results of the currently implemented polling rate, the maximum polling rate, optimized B-SIPS static solutions, and the B-SIPS dynamic solution.

4.5 Conclusions

This chapter explored the alteration of the smart battery polling rate as a means to increase the device security and lifetime. First, a background on smart battery transmission details was explored. Next, static optimizations were performed for both simulated and realistic networks were performed; the results presented in this section of the paper were published and presented in the 2007 IEEE Southeast Conference [7]. Finally, a dynamic solution was devised and simulated. Results pertaining to this dynamic solution were published in the Eighth Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop [6].

The dynamic solution provided the most positive impact on the lifetime of a device running B-SIPS. The extent of the increased lifetime can be seen in Figure 12, which shows the increased percentage of life afforded by static and dynamic polling solutions [7]. Note that the percentage increase for static solutions is based on the polling rate optimized for that specific network density; the static lifetime increase percentages do not all belong to a single smart battery polling rate. The lifetime increase percentage increase for the dynamic solution represents an average of the percentage increases from clustered and distributed attacks.

Figure 12. Dynamic Polling Impact Percent Lifetime

Attack Density	Implemented OEM (1Hz)	Max. Life % Increase for Static Polling	Ave. Life % Increase for Dynamic Polling
0	12.4010	2.346584953	2.23
1	12.4000	2.088709677	2.08
10	12.3920	1.565526146	2.14
100	12.3050	0.43071922	2.82
1,000	11.4400	2.001748252	10.10
10,000	2.7919	251.0942369	331.09
20,000	0.0595	14,850.64158	**19,094.29 †**
40,000	0.0595	13,000.07904	17,001.14
60,000	0.0595	11,800.34139	14,908.07
80,000	0.0595	10,896.41794	12,815.09
100,000	0.0595	9,992.494492	10,722.11

† Denotes theoretical maximum increase attainable with dynamic polling.

The cost of this security and lifetime improvement is that of hardware complexity. In order for these improvements to come to pass, future smart batteries will need to support bidirectional communication on their SMBuses, as well as allowing real-time reconfiguration of smart battery data rates. As an intermediate solution, however, smart battery manufacturers can easily implement an optimized smart batty polling rate for use with B-SIPS by changing the hard coded polling rate somewhere between 8.33 Hz and 25 Hz. Regardless of the path taken, the alteration of smart battery polling rates, according to the research explored in this chapter, has the capacity to increase the effectiveness of B-SIPS, as well as the security and lifetime of mobile devices.

5. B-SIPS NETWORK SIMULATIONS

This research creates and analyses large scale network simulations as a means of determining the likelihood of success in B-SIPS deployment endeavors. The network simulations extend research completed by Buennemeyer, who suggested that the use of network simulations should be investigated as a means of validating the feasibility of B-SIPS application deployment [8]. The network simulator OPNET is used to determine the burden that external applications experience due to the deployment of B-SIPS. Once this *service degradation*, or the reduction of an application's average data reception rate, has been determined, deployment optimization suggestions can be made. The remainder of the chapter will be structured as follows: Section 6.1 will review the assumptions and structure of the simulation topology and Section 6.2 will discuss hypotheses made pertaining to optimum B-SIPS deployment implementations. Section 6.4 will review the network simulations created and executed using OPNET. Finally, Section 6.5 will summarize the results and findings of the chapter.

5.1 Simulation Topology

Assumptions and simplifications must be made when creating simulations, as modeling precise networks would be cumbersome and inefficient. This section outlines the assumptions made, as well as simplifications which made our simulation more general. The first assumption dictated that the simulation deployment was to be applied to the network infrastructure on the Virginia Tech campus. Specifics regarding core routers, building access points, link capacities, access point capacities, and general network topologies were obtained [11]. A *core router* is a router on the edge of an internal network, which connects that internal network with an external network. *Access Points* allow wireless mobile devices to communicate with one another, as well as with the wired network. Finally, n*etwork topologies* specify the hardware layout of a network. Examples of network topologies can be seen in Figure 13, Figure 14, and Figure 15.

Once this information was obtained, the spectrum of the simulation needed to be defined. In order to study the throughput of applications being used by students, faculty, and staff in academic buildings, the access points in those academic buildings needed to be modeled. Since the link capacities between mobile devices and these access points is much lower than the link capacities between the access points and the servers that the mobile devices are trying to access, the study was able to be reduced to a smaller scenario that contains only two academic buildings on campus. We chose the buildings Whittemore and Durham Halls to model, since these two buildings cater primarily to those who belong to the Bradley Department of Electrical and Computer Engineering. Assumptions were made for each of these buildings such that the number of access points on each floor is consistent, and such that the access points were evenly distributed. Whittemore Hall has five floors with a total of 31 access points, and was therefore simplified such that each floor would contain six evenly distributed access points. Durham Hall has four floors with 29 access points, and was simplified into a scenario where each floor contained seven evenly distributed access points. Figure 13 depicts the topology of a floor in Durham hall. The round disks represent access points and the blue computers represent the mobile devices running B-SIPS. The access points are assigned an operating channel of 1, 4, 7, or 11, and these channels are spaced out amongst the access points in such a manner as to avoid as much interference as possible. Mobile device placement details will be addressed once the wired portion of the network has been defined.

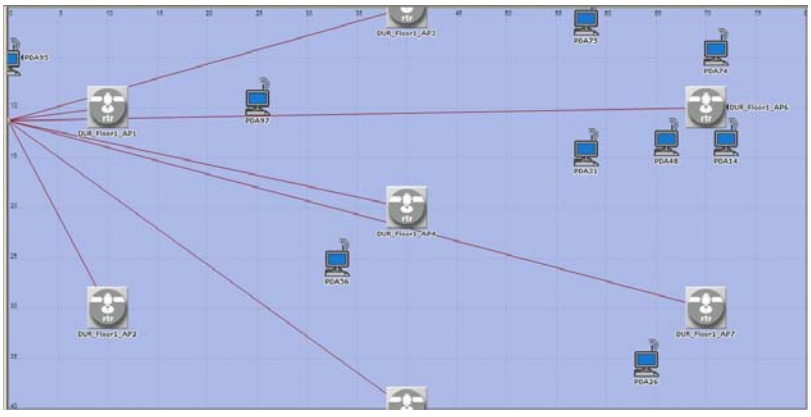


Figure 13. OPNET Simulation's building floor topology

Note that the access points are each connected to an undetermined point. The point is external to the subnet seen in Figure 13, and is more clearly depicted in Figure 14. It connects the wired interfaces of each access point in a floor with the building floor's switch. Each floor's switch is then routed to the building's switch. All connections thus far have a link capacity of 100 MB.

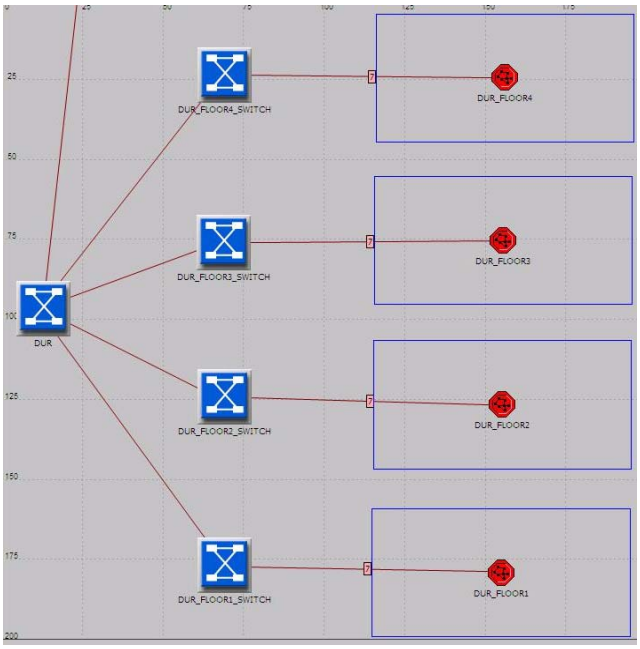


Figure 14. OPNET Simulation's building topology

Next, each building in the network is connected to one of the university's core routers, as seen in Figure 14. Due to the topology's building selection, Whittemore and Durham, the building switches are connected to the Burruss core router. This connection has a link capacity of 1 GB. Finally, the core router is connected to the B-SIPS' CIDE server, and well as networks external to the university, which include, but is not limited to, mail service providers, search engines, and FTP servers. For our simulations, all of these possible connections are simplified into a single server, which is denoted FTP_server in Figure 15, and is also simplified in such a manner that it is directly connected to the core router, rather than connected via an indirect path. The core router's connections to both the internal B-SIPS' CIDE_Server and the external FTP_Server have a link capacity of 1 GB.

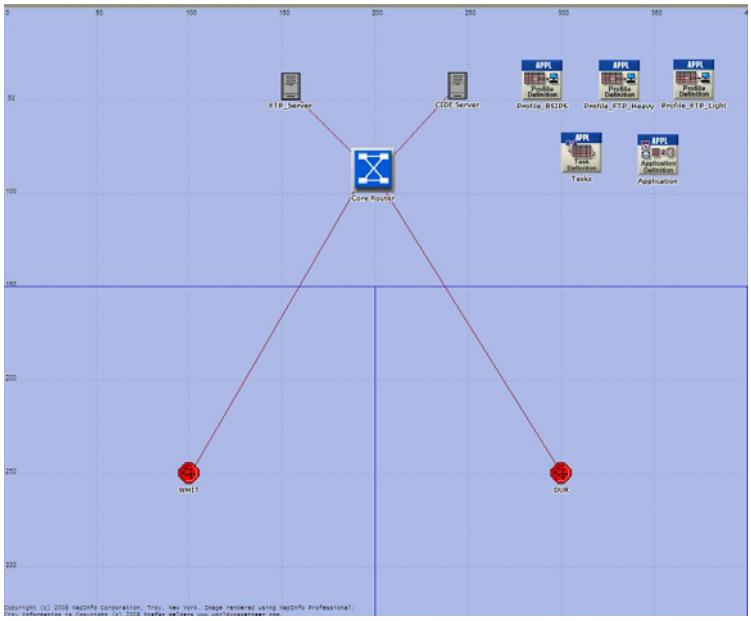


Figure 15. OPNET Simulation's campus topology

Once the completion of the wired portion of the network simulation topology was complete, wireless mobile devices were defined. Due to the necessity to test B-SIPS in a variety of network scenarios, simulations would need to be run using different network sizes. In this regard, the network size refers to the quantity of mobile devices connecting to the available access points. The maximum network size was determined to be approximately 1800 mobile devices, and was derived by multiplying the number of access points present in the simulated topology, 60, with the suggested maximum capacity of a single access point, which is 30 mobile devices [11]. Simulations were performed for network sizes of 100, 200, 300, ..., 2000 to ensure that the effect of B-SIPS was tested in device unsaturated and device saturated scenarios.

To determine the topology of these devices, a Perl script, as seen in Figure 16, was devised. The script creates the simulation topology by using a random number generator to determine the building, floor, and (x,y) coordinates for each device. Once locations are determined, each mobile device is placed in the appropriate subnet and assigned to the closest access point.

```
 #!/usr/bin/perl -w

use strict;

########################################
# Building 1 = DURHAM HALL       #
# 4 floors high, 29 Access Points #
# x = 80m, y = 40m              #
########################################
my $building1         = 0;
my $building1_floors   = 4;
my $building1_Xmax     = 80;
my $building1_Ymax     = 40;

########################################
# Building 2 = WHITTEMORE HALL    #
# 5 floors high, 31 Access Points #
# x = 90m, y = 30m              #
########################################

my $building2         = 0;
my $building2_floors   = 5;
my $building2_Xmax     = 90;
my $building2_Ymax     = 30;

####################################
# Select positions for all PDAs #
####################################
my $pda_num   = 0;
my $num_pdas  = 2000;
my $bldg_full = "true";
my $ap_full   = "true";

while ($pda_num<$num_pdas){

   ##########################
   # Pick a random building #
   ##########################
   my $random_building = int(rand(2));


            ############################
            #######
   # Pick a random floor & position #

            ############################
            #######
   my $building_name = "";
   my $random_floor  = 0;
   my $x_position    = 0;
   my $y_position    = 0;
   if ($random_building eq 0){
            $building_name = "DUR";
            $random_floor  = int
        (rand($building1_floors));
            $x_position    = int
        (rand($building1_Xmax));
            $y_position    = int
         (rand($building1_Ymax));
   }else{
            $building_name = "WHIT";
            $random_floor  = int(rand
         ($building2_floors));
            $x_position    = int(rand
         ($building2_Xmax));
            $y_position    = int( rand
         ($building2_Ymax));
   }

   printf "PDA: $pda_num\tBuilding:
            $building_name\t\tFloor:
            $random_floor\t\tPosition:
            ($x_position,\t$y_position)
      \n";
   $pda_num++;
}
```

Figure 16. Mobile Device Location Random Generator

With the wired and wireless portions of the network simulations defined topologically, the simulation assumptions section is complete. An explanation of how the topologies defined here will be tested, as well as the expected outcome of those tests, will be reviewed in the following section of this chapter.

5.2 Simulation Tests & Hypotheses

Determining the affect the B-SIPS application has on the network in which it is employed requires testing the throughput of network while they are both using, and not using the application. The evaluation of B-SIPS is therefore judged on the degradation of service, both of the B-SIPS applications and external applications, which occurs when B-SIPS is run in conjunction with resource-hungry applications.

Mobile devices in each scenario are simulated running the B-SIPS application only, a File Transfer Protocol (FTP) client only, and both the B-SIPS application and FTP client to measure this degradation. Once these simulations are complete, Equation 2 will be applied:

Equation 2. Throughput Degradation Determination

$$Degradation = \frac{Application\_Throughput(Baseline) - Application\_Throughput(Test)}{Application\_Throughput(Baseline)} * 100$$

Simulations and degradation calculations were performed for each network simulation size, and additionally, using varying B-SIPS transmission rates for each of the aforementioned simulation sizes. B-SIPS applications send 500 bytes of data once per second, by default. By altering the transmission rate of the application, B-SIPS could instead transmit a 500*x* byte packet and then not transmit again for *x* seconds. As the small mobile devices only poll their smart batteries once per second, there would be no point in transmitting data to the CIDE server more often than the default transmission rate. Along the same lines, the application developers requested that data not be transmitted less frequently than once per minute, as repercussions due to lost packets, due to short bursts of data on the network, become quite disruptive when an entire minuet's worth of data is sent in a short period of time. Therefore, simulation tests encompass network sizes of 100 to 1500, in increments of 100 mobile

devices, and transmission frequencies used by the B-SIPS application are defines as once every 1, 10, 20, 30, 40, 50, and 60 seconds.

The first hypothesis made is that the B-SIPS and FTP applications' will both experience degradation, and that their degradations will react oppositely to a decrease in B-SIPS transmission period. This hypothesis is due a heightened probability of packet collisions as the size of the B-SIPS datagram increases. This phenomenon should indicate that when the B-SIPS packets are small and frequent, they are less likely to be stepped on by external applications' transmission. Those external applications, however, may be more likely to be stepped on, as interruptions for B-SIPS will be occurring frequently. Conversely, as the transmission period declines, this hypothesis states that the degradation of the external application will decrease and the degradation of the B-SIPS application will increase. This phenomenon should cause the graph of application degradation values to have an intersection of lines.

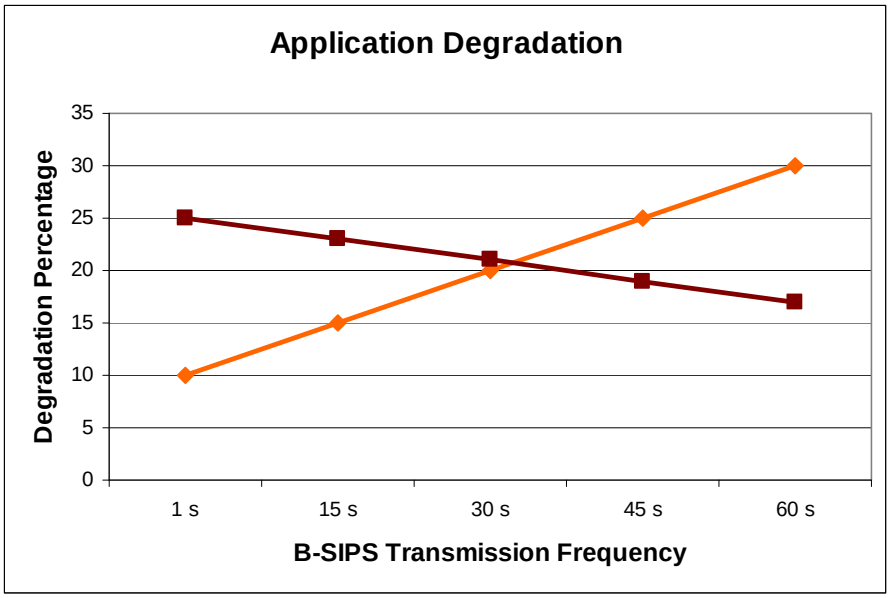


Figure 17. Degradation Caused by B-SIPS

The second hypothesis is that there will be an optimum transmission period, which acts as a compromise between degradation of the B-SIPS application and degradation of external applications. A graph of transmission frequencies and degradation percentages will indicate this optimum value, or *break even point*, as the intersection of the lines that represent the B-SIPS and external applications (Figure 17). Finally, the third hypothesis states that the break even point associated with a network will increase as the number of mobile devices increase, as shown in Figure 18.

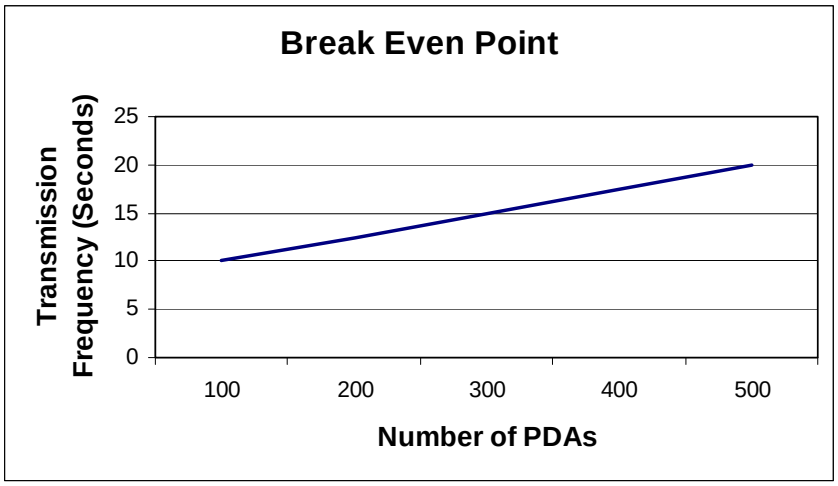


Figure 18. Break Even Point for Variously Sized Networks

5.3 OPNET Simulations and Results

Ns-2 proved inadequate to handle the B-SIPS large scale simulation initiative. OPNET was selected as the simulation tool. Section 5.3.1 provides an overview of simulation construction overviews, and Section 5.3.2 presents simulation results and draws deployment conclusions for the B-SIPS client and server.

5.3.1 Construction of Simulations

This section presents the construction of B-SIPS simulations using OPNET; all information regarding configuration definitions and practices was obtained from the product documentation and tutorials, found in the *help* menu option of the OPNET simulation tool [25]. As mentioned in Chapter 2, OPNET is a GUI based application, which provides the user with base objects that are configurable. The most important of these objects, in respect to B-SIPS simulations, are the *application*, *profile*, and *task* configuration objects. These objects, as shown in Figure 19, are placed in the highest level of the simulation diagram, which was depicted as the Campus Network Topology level in Section 6.1.

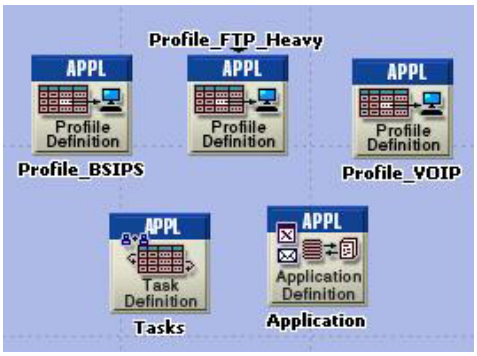


Figure 19. Application, Profile, and Task Configurations

The application configuration object helps to determine the applications that will be made available to each of the hardware devices in the simulation. Default applications included in OPNET include low and high load database access, email, file transfer protocol (FTP), file print, telnet session, video conferencing, voice over IP call (VOIP), and web browsing. For each default application that is desired to be usable by simulation devices, an associated profile must be created. Note that the B-SIPS simulation uses three different profiles. As shown in Figure 20, each of these profiles dictates the applications associated with the profile, the application start time, as well at the duration and the repeatability of the application. The application *start time* refers to the offset associated with the application's profile. If it were desired for a simulation to show phases of varying network traffic, the offset value may be set to a non-zero value. For the B-SIPS simulation, however, all profiles have a start time of 0 seconds. The *duration* is the maximum time allotted for the application session, which is

generally used as a timeout mechanism. As this simulation initiative does not intent on foreshortening any application delay, which would skew throughput data results, the duration is simply set to "End of Last Task". Finally, the *repeatability* of an application determines whether a transaction occurs a single time, or throughout the duration of the simulation. For all B-SIPS simulation profiles, the applications possess unlimited repeatability.

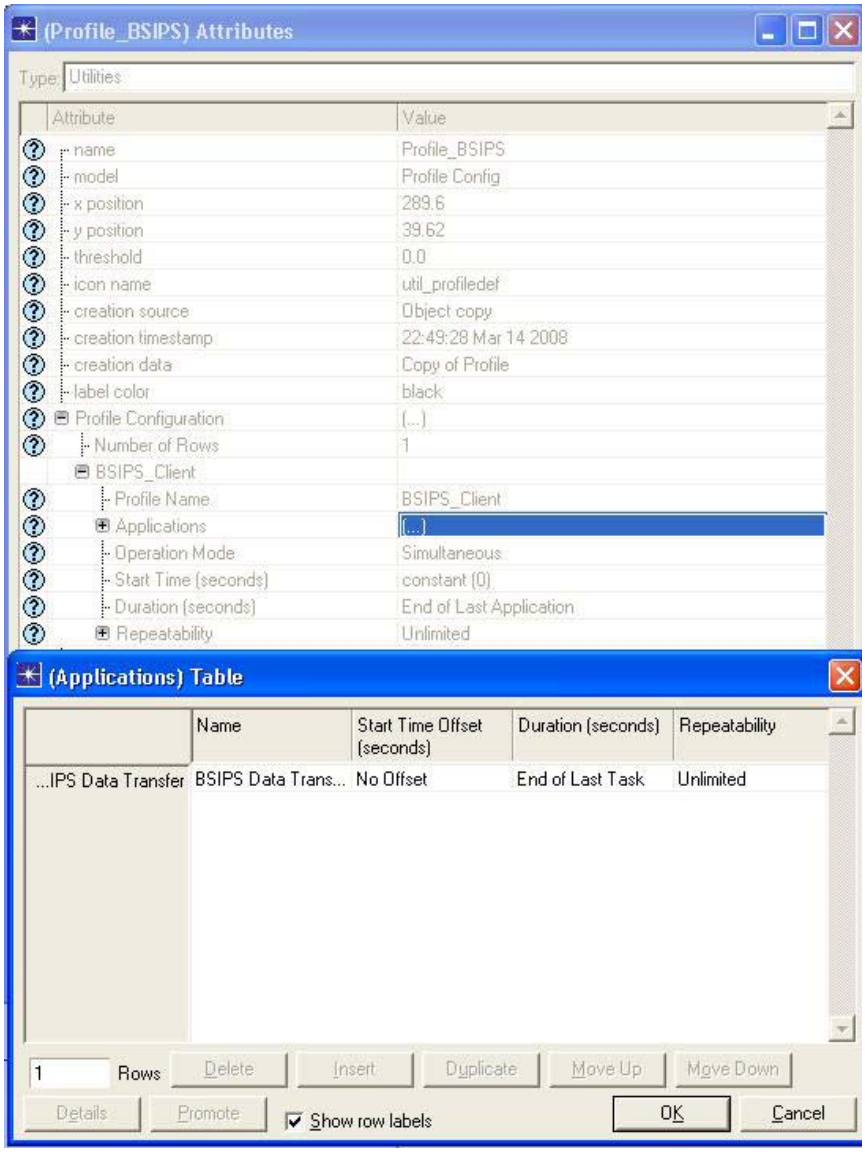


Figure 20. Profile Configuration

Profiles designed to run custom applications, such as B-SIPS, are slightly more complicated due to their use of the *task* object. Tasks allow users to schedule specific client / server interaction, which can be catered to model a particular custom application. To do so, the task attributes window must be brought up and the task specification section edited appropriately (see **Error! Reference source not found.**).

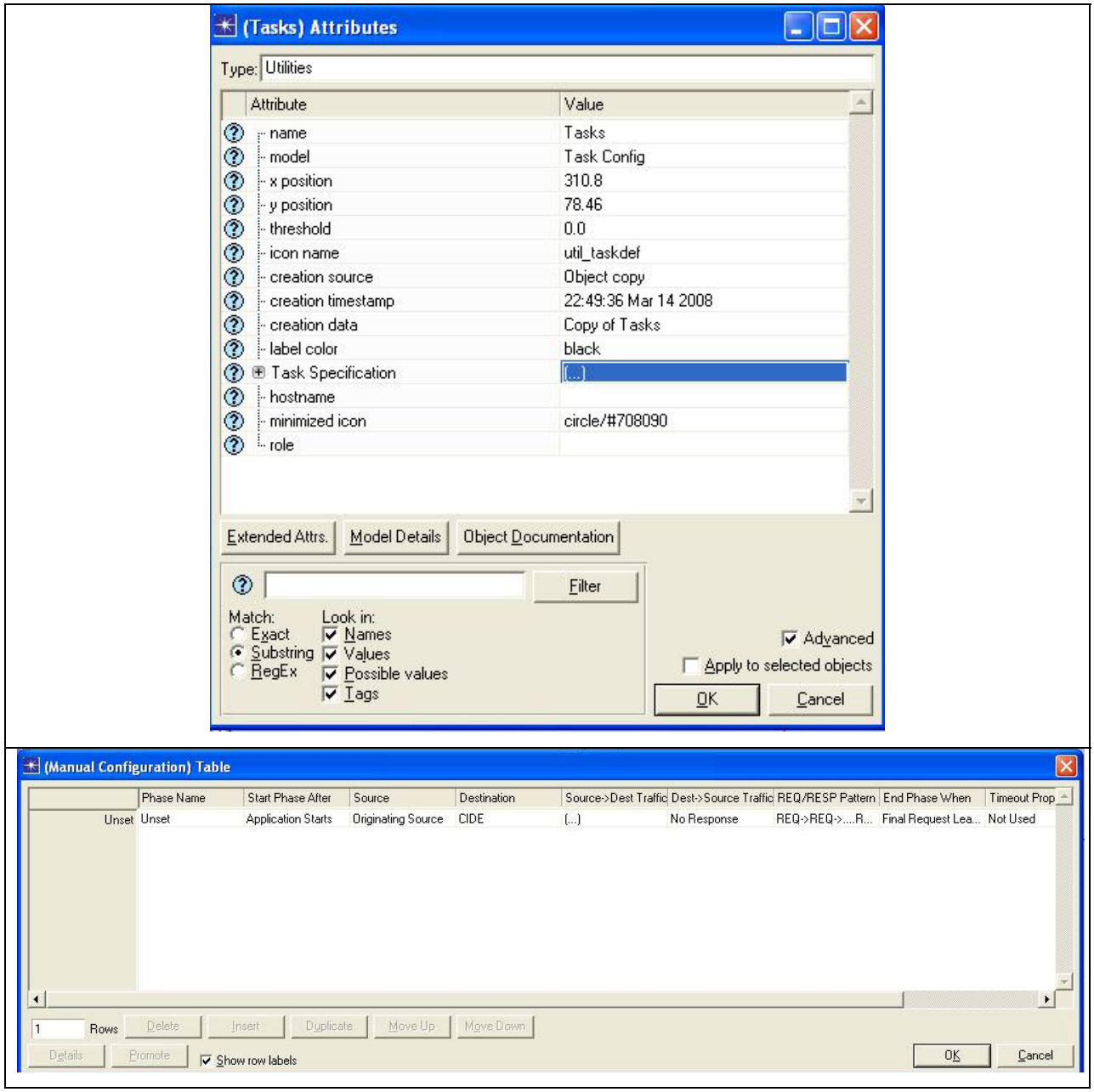


Figure 21. Task Configuration

The only field in the Task Specification Table that is important for our purposed is the Manual Configuration field, which after being clicked, brings up the Manual Configuration Table window. This table requires only a single row, as the B-SIPS application transmits UDP datagrams and does not expect any returning data. The single phase begins as soon as the application starts. The *source* of the application is the device where the profile containing the task is running. The *destination* will be the B-SIPS' CIDE server. The *Source -> Dest Traffic* will open up an addition window (Figure 22).

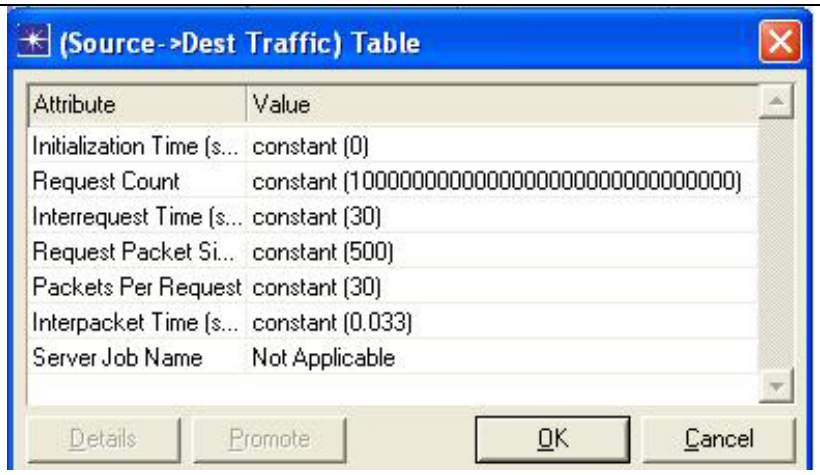


Figure 22. Task Configuration – Source and Destination Traffic Table

This window specifies the communication between the source and destination devices. The *Initialization Time* is an offset, in seconds, and is set as zero. The *Request Count* is the number of requests the task will attempt, and is set to an extremely large number, such that requests will occur throughout the entire simulation. The *Interrequest Time* is the delay between the end of one request and the beginning of the subsequent request. This value will vary with different B-SIPS simulations. The *Request Packet Size* is set to 500 bytes, which is the size of a B-SIPS packet. The *Packets Per Request* is set to 1 if the application transmission period is once per second, and increases as the period decreases. This is variable, rather than the packet size, because RFC 3226 suggests that applications do not transmit packet fragments larger than 512 bytes [15][13]. Due to the fragmentation of application data

packets several transmissions are required in order to send application data exceeding 512 bytes. The delay experience between the transmissions of these packet fragments is termed the *Interpacket Time*.

The FTP application was chosen as the application to accompany B-SIPS in the network simulations, with the intention of determining how file transfers are affected when B-SIPS security data is being transmitted from a mobile device simultaneously. We simulate a high load FTP application due to the large capacity of network throughput that file transfers typically require. To accomplish the simulation of FTP and B-SIPS on all mobile devices in the network topology, all devices are configured, as shown in Figure 23.

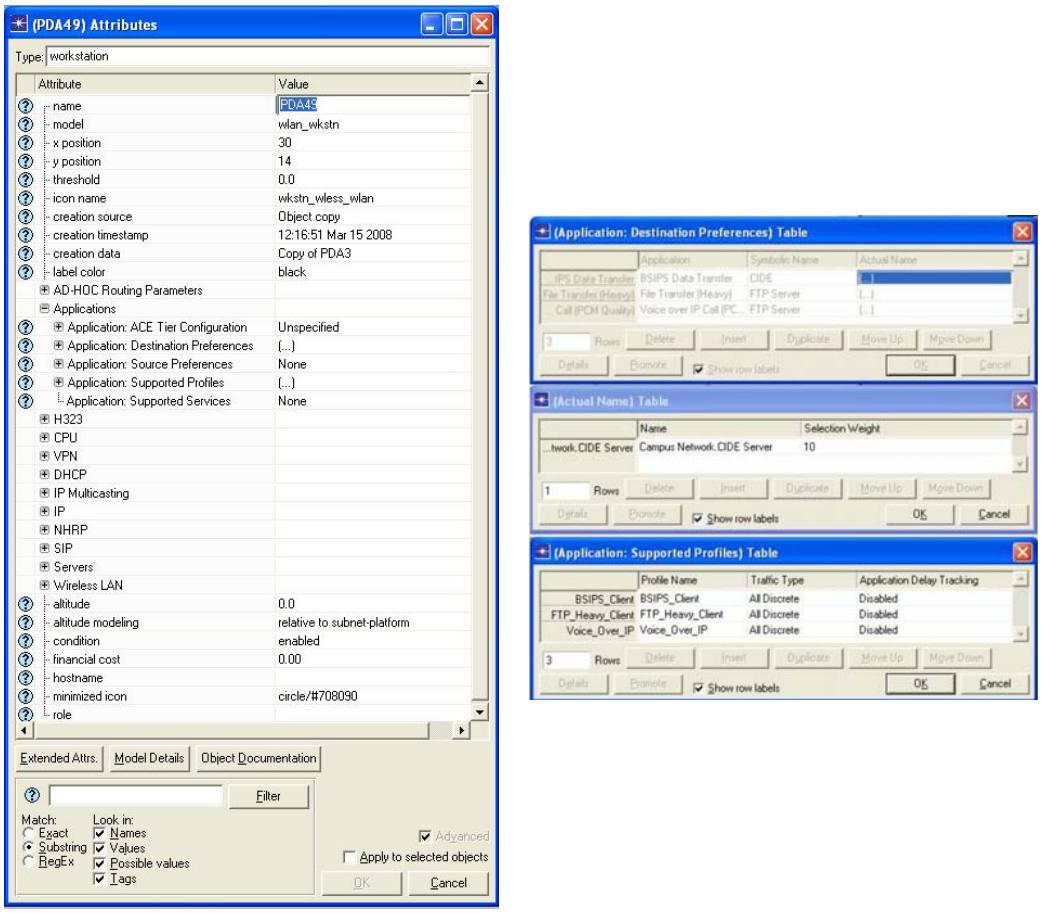


Figure 23. Configuring Devices

Within the PDA attributes, the section "Applications" is expanded and "Supported Profiles" is selected for editing purposes. In OPNET, selecting a *profile* for a network device is synonymous with dictating that the device will be running a client of the application specified by that profile. In order to run an FTP client, the number of rows in the Supported Profiles table will be set to 1 and the profile name will be set to FTP_Heavy_Client. Next, the "Destination Preferences" row in the PDA Attributes window is selected. This determines where each running application will be destined towards during simulations. The application is set to "File Transfer (Heavy)", as defined by OPNET. Seven applications are provided for OPNET simulators' utilization, where each application has network load options of "Light" and "Heavy". The symbolic name to FTP Server is set to "FTP Server", which matches the naming scheme used in the topology presented previously. The field "Actual Name" opens up a new window. A single row is added to the table and "Campus Network FTP Server" is selected from the dropdown menu. Finally, the FTP server is edited such that its application attributes include "File Transfer (Heavy)" in the "Supported Services" section. Similar modifications are made for scenarios in which mobile devices are running B-SIPS.

Configuration modifications are complete. Desired attributes to be tracked include the transfer and reception rate (bytes/sec) of both the FTP and B-SIPS applications. Results and Conclusions are introduces in the following section of this chapter.

5.3.2 Results & Conclusions

OPNET simulations were run on a Dell Optiplex desktop computer running Windows XP and containing a 2.80 GHZ processor and 2 GB of RAM. Desired network simulations of 1500 mobile devices, or fewer, were easily completed. Simulations containing network sizes between 1600 and 2000, however, required an additional 2GB of memory and Windows boot file alterations in order to enable OPNET to allocate more than 2GB of RAM. Once this was accomplished, the remaining simulations were completed successfully.

The initial simulation testing done on each network size incorporated mobile devices running both a heavily loaded FTP client and the B-SIPS application, with a B-SIPS data transmission rate of one packet per second. Results pertaining to these scenarios are shown in Table 6. The reception percentage is calculated by dividing the average reception rate by the average transmission rate for each application. As shown in Table 6, there is no downward trend for reception percentages. This disproves our first hypothesis and invalidates the remaining subsequent hypotheses. The reception percentage for the FTP client application, which utilizes the throughput greed TCP transmission protocol, does not dip below 99%. The B-SIPS client application, however, is uses the UDP transmission protocol, and therefore has a slightly lower reception percentage. The minimum B-SIPS application reception percentage dips down to 97%. The reception percentages of each application was high enough to make the conjecture that under normal to heavy network usage, the data transmitted by the B-SIPS application did not make a substantial impact on network throughput.

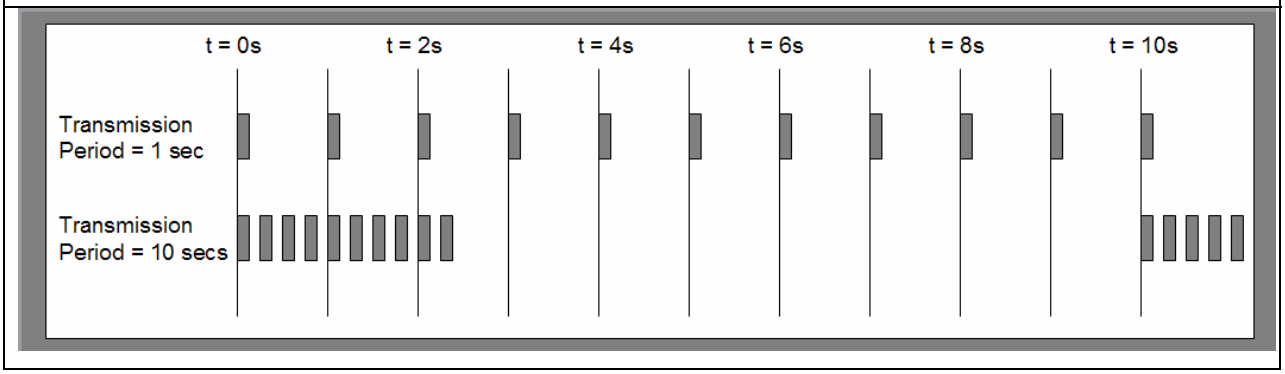
Table 6. Simulation Reception Percentages

Number of Mobile Devices	B-SIPS Reception Percentage	FTP Reception Percentage	Number of Mobile Devices	B-SIPS Reception Percentage	FTP Reception Percentage
100	100.00000	99.99994	**1100**	98.31580	99.99988
200	99.99985	100.00000	**1200**	99.99990	99.99990
300	98.72440	99.99996	**1300**	100.00000	99.99997
400	98.56610	99.99983	**1400**	97.79776	99.99996
500	99.79696	99.99986	**1500**	97.40660	99.96257
600	100.00000	99.99991	**1600**	97.40660	99.96257
700	99.21957	99.99995	**1700**	99.96847	99.99994044
800	97.53378	99.99987	**1800**	99.73847	100
900	100.00000	100.00000	**1900**	99.94467	99.96608295
1000	100.00000	99.99988	**2000**	99.97427	99.99992932

Prior to the affirmation of this conjecture, however, secondary simulation testing was embarked upon, as a means to validate the custom OPNET model of the B-SIPS client application. In addition to the heavily loaded FTP and B-SIPS applications, each mobile device in the network ran a heavily loaded voice over IP (VOIP) application. VOIP is a UDP based program that utilizes the wireless network, rather than standard cellular network. This was chosen as the secondary external application not only for the heavy load it inflicts on the network, but also for the popularity of the application, which is used heavily for international communication. This is the technology used by applications such as Google Talk and Skype. Additionally, since VOIP uses UDP and FTP uses TCP, the affect of B-SIPS on resource saturated networks is determined for both TCP and UDP external applications.

Simulations containing the transfer of such large amounts of data naturally consume a large portion of system resources, and are extremely slow. To prove the validity of the B-SIPS application, it was therefore neither necessary nor worthwhile to run simulations for every network size. Simulations were run for network sizes of 100, 200, 300, and 400 mobile devices, with B-SIPS transmission periods of 1, 10, 20, 30, 40, 50, and 60 seconds. Note that although the B-SIPS transmission period may vary between simulation scenarios, the total quantity of data being transferred between the B-SIPS client and server should remain stable (see Figure 24).

Figure 24. B-SIPS Application Transmission Period



The transmission period merely dictates what the quantity of data is that is transferred. When the transmission period is set to 1 second, a single B-SIPS application packet is transmitted every second, whereas when the transmission period is set to 10 seconds, 10 B-SIPS application packets are transmitted every 10 seconds. The aim of altering the transmission period in the OPNET simulations is to determine whether transmitting single B-SIPS application packets or clusters of application packets is more efficient. For each simulation scenario that is created by combining a network size and a B-SIPS transmission period, one simulation run shows the B-SIPS throughput baseline, one shows baseline for combined FTP and VOIP traffic, and a final simulation shows the throughputs from when all applications run synchronously.

The first portion of collected data which will be examined pertains to the B-SIPS application. Equation 3 depicts the manner in which throughput degradations for B-SIPS were calculated. The B-SIPS baseline throughput is defined for each scenario to be the average throughput, in bits per second, when B-SIPS is the only application running on the simulated mobile devices. The B-SIPS test throughput is defined as the average throughput for the B-SIPS application while the mobile devices are running B-SIPS, FTP, and VOIP.

Equation 3. B-SIPS Degradation

$$\frac{(BSIPS\_Baseline)-(BSIPS\_Test(All\_Applications))}{BSIPS\_Baseline}*100$$

These throughput values are used to determine the percent degradation in B-SIPS caused by the application being run in conjunction with FTP and VOIP clients. The degradation percentages for each of the simulated scenarios are shown in Figure 25.

Figure 25. B-SIPS Degradation in High Load Networks

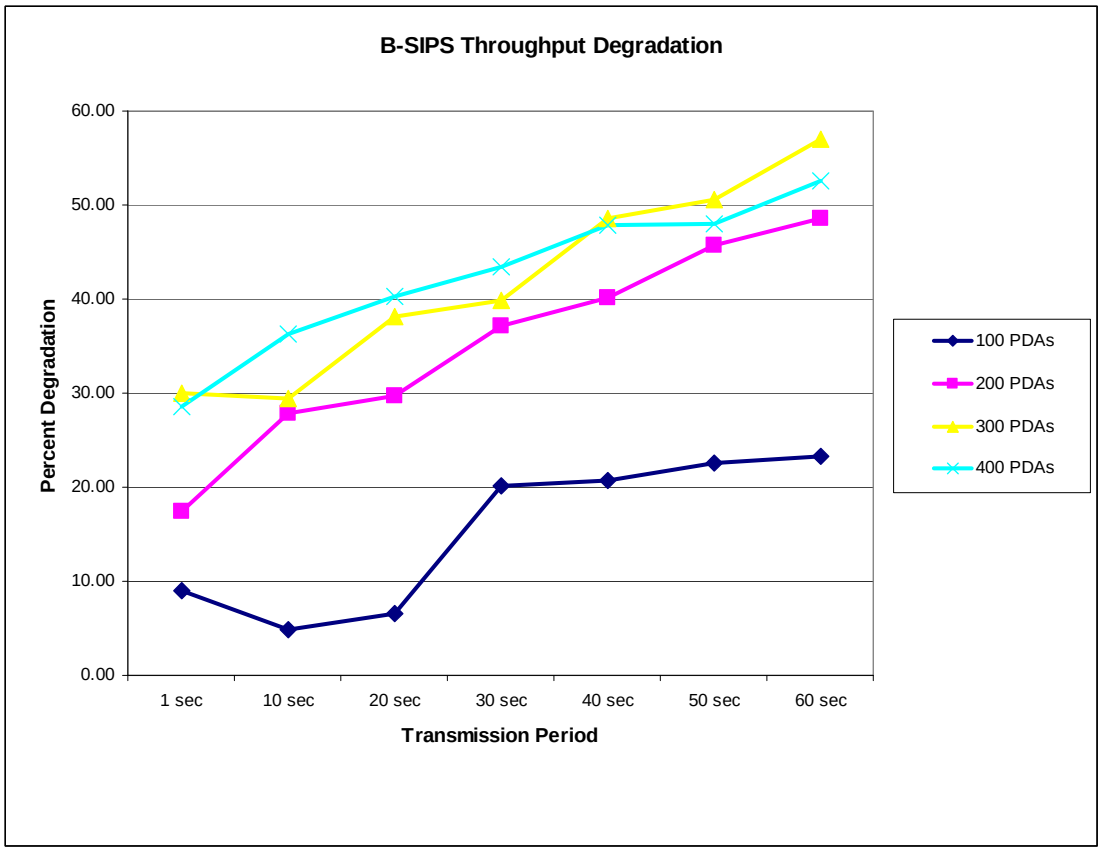


Figure 25 shows that when the B-SIPS application is thrust into a load saturated network, via the addition of FTP and VOIP client applications, increasing the network size will in turn increase the application throughput degradation. This observation is intuitive. Additionally, we find that as the degradation of the throughput increases fairly linearly with B-SIPS application transmission period. This observation is also straight forward, though data pertaining to a network size of 100 mobile devices and an application transmission period of between 10 and 20 seconds does not seem to adequately fit this model. This inconsistency might be caused by a number of things, including the selected topology configuration, which may have allowed small clusters of data from B-SIPS to be transmitted with little interference, due to the extremely sparse nature of the topology. Larger clusters of data, however, may have caused more of a disruption to the throughput of B-SIPS. As a general

trend, however, Figure 25 shows that the most effective transmission period for successful B-SIPS applications throughput is the default frequency of 1 Hz, or a period of 1 second.

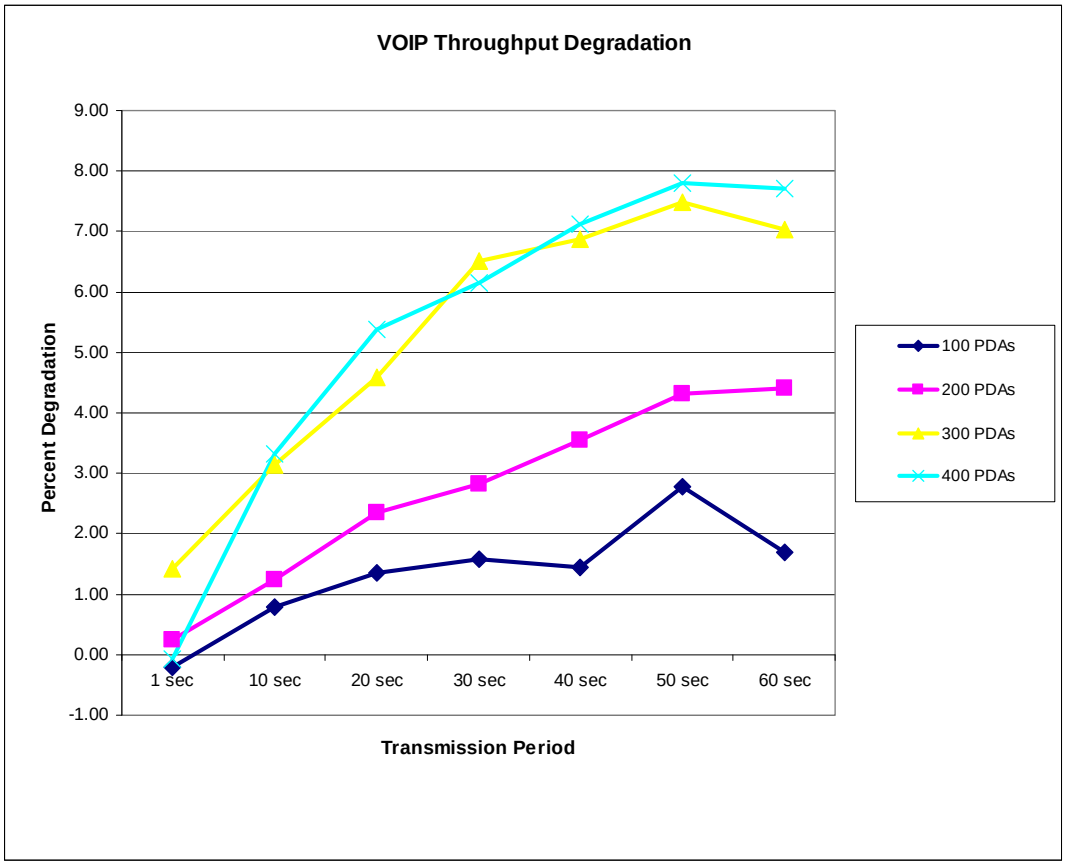
Next, we examine the throughput degradation associated with external UDP applications, modeled in these simulations by the VOIP client. Calculations performed for this investigation are presented in Equation 4, as differ slightly in theory from the equation that was utilized for B-SIPS degradation. The VOIP baseline is defined as the VOIP application throughput when all simulated mobile devices are running both the VOIP and FTP client applications. This is the case because we are trying to determine the effect of B-SIPS on applications that are already running in load saturated networks. The VOIP test throughput is defined as the throughput of the VOIP application when all applications, including B-SIPS, VOIP, and FTP, are deployed and running on each of the simulated devices.

Equation 4. VOIP Degradation

$$\frac{(VOIP\_Baseline(VOIP\_FTP)) - (VOIP\_Test(All\_Applications))}{VOIP\_Baseline(VOIP\_FTP)} * 100$$

Application degradation for VOIP, which models external UDP applications, running in highly saturated networks is presented to the reader in Figure 26. Results shown here are similar to those that depicted the B-SIPS application degradation in that the VOIP application degradation is shown to increase proportionally with both the network size and transmission period.

Figure 26. VOIP Degradation in High Load Networks



Again, however, there are a couple of discrepancies to note. The first is seen when we examine the percent degradation associated with a transmission period of 1 second. Note that network sizes of 100 and 400 PDAs actually have a negative degradation for this transmission period. This could be due to OPNET rounding errors, as the value is only slightly below 0% degradation, or this phenomenon could be the result of the VOIP application utilizing the throughput that has been degraded for either of the other applications (B-SIPS and / or FTP). Additionally, we note that the degradation associated with an application transmission period of 60 seconds seems to approach a decreasing trend. This may indicate that transmission periods greater than 60 seconds will lend lower external application degradation values than those experienced for transmission periods of 1 second. However, due to a

constraint by the development team of B-SIPS that states the B-SIPS security data must be transmitted at least once every minute, this possibility was not an endeavor worth exploring. Additionally, we know from previously explored data that B-SIPS application transmission periods greater than 60 seconds will cause detrimental degradation for B-SIPS security data. Therefore, the optimum transmission period with respect to the VOIP application is the default transmission period of 1 second.

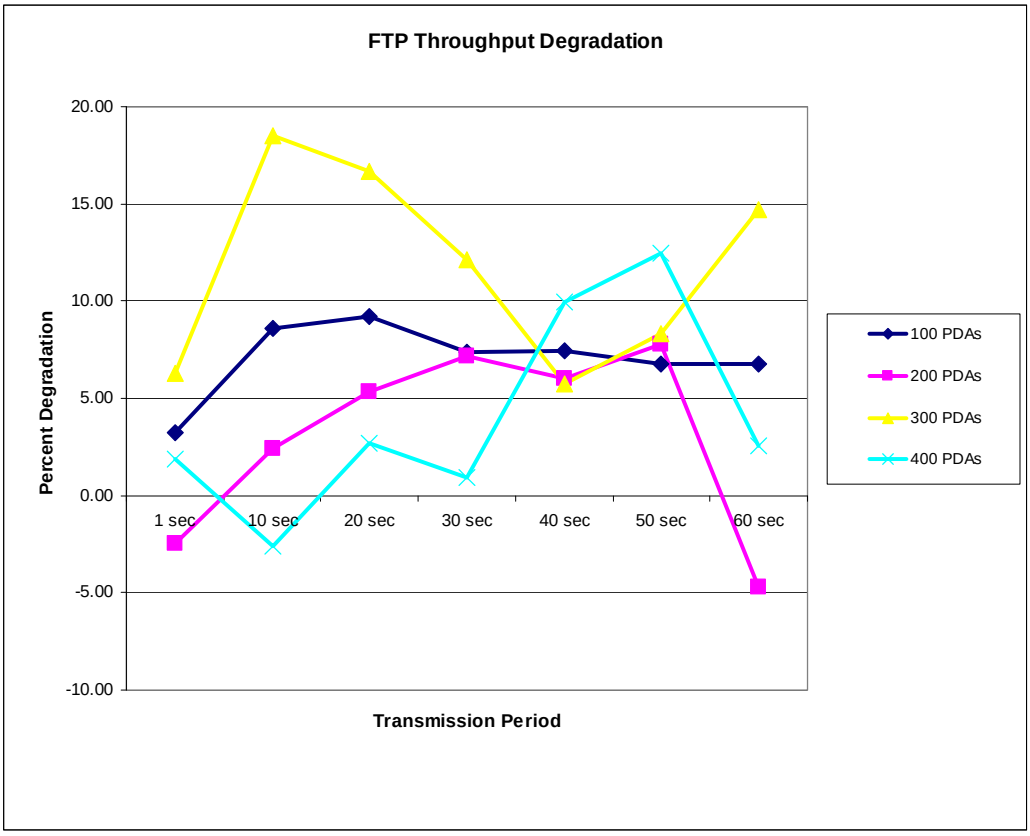
The remaining simulation data to investigate pertains to the application degradation of the FTP client. This data will depict the effect of the B-SIPS client application on external TCP applications that are running in load saturated network conditions. Degradation calculations are shown in Equation 5, and mirror those calculations performed on the VOIP application, save that the baseline and testing throughputs used are in respect to the FTP client application, rather than the VOIP application.

Equation 5. FTP Degradation

$$\frac{(FTP\_Baseline(VOIP\_FTP)) - (FTP\_Test(All\_Applications))}{FTP\_Baseline(VOIP\_FTP)} * 100$$

Information parsed from this degradation equation is shown graphically in Figure 27. As seen immediately upon investigation, this data does not at all follow the degradation trends of the previous two applications. In fact, the data does not seem to follow much of a trend at all, save for the fact that the application transmission period with the smallest cumulative degradation percentage, with consideration taken for all simulated network sizes, is an application transmission period of 1 second.

Figure 27. FTP Degradation in High Load Networks



The reason for the unpredictable nature of the remaining data is unknown, however, reasonable conjectures can be made. The most plausible explanation deals with OPNET's calculation of throughput for TCP and TCP-like protocol applications. It is unknown whether data packets are the only type of packet that is included in their calculations, or whether retransmitted and acknowledgement packets are also considered. If so, this may be cause the sporadic nature of the FTP data, additionally rendering the collected FTP throughput data inadmissible for transmission period optimizations. Alternatively, the unpredictability of the FTP throughput degradation may be caused by some other phenomenon. Further investigation of the effect of B-SIPS on TCP applications is suggested for rising researchers in the network security field. Therefore, we will consider the FTP data results solely to for their aid in validating the OPNET application model of B-SIPS.

In summary, the network simulator OPNET was used to simulate the B-SIPS application, in conjunction with both FTP and VOIP applications. It was shown that when B-SIPS ran in mobile devices simultaneously with a solitary FTP client, the throughput remained exceptional, as experienced by both applications. Additionally, a second simulation showed that while applications running in saturated networks experienced degradation, this degradation could be minimized by setting the transmission frequency to 1 Hz. Due to the completion of the secondary round of simulations, the initial conjecture holds. Simulation results pertaining to the Virginia Tech network support the claim that B-SIPS is a reasonable solution for deployment, and therefore, it is believed that networks with similar topologies will also experience success in this endeavor.

5.4 Summary

This chapter explored the large scale simulation of the B-SIPS application in an endeavor to determine the soundness of network deployments of B-SIPS, in terms of the throughput degradation of externally running applications. The topology was constructed such that the network simulation reflected a portion of the Virginia Tech wired-cum-wireless network. Additionally, hypotheses were made stating that degradation due to B-SIPS was expected, as were break even points that would be represented as points at which the graphed degradation lines for B-SIPS and FTP would cross. Next, OPNET was used to construct two simulations, which showed that networks containing resource-greedy external applications were not adversely affected. Additionally, it was shown that highly saturated networks did experience throughput degradation due to B-SIPS, but that the degradation was not detrimental, and could be minimized by using the default transmission frequency of 1 Hz. Therefore, the conclusion drawn by this chapter is that B-SIPS is a viable application for network deployment, and operates optimally when used with its default settings.

6. CONCLUSIONS AND FUTURE WORK

The improvements and validations performed in this research were geared towards promoting the deployment of B-SIPS for small mobile devices around the world. Areas of focus in the stated validation included ensuring public acceptance of the application through the implementation of a usability study and verifying that the deployment of the application would not jeopardize the performance of external mobile device applications. Additionally, GUI optimizations were realized for both the B-SIPS client and CIDE server, future smart battery hardware implementations were introduced for increased effectiveness with the B-SIPS application, and optimum deployment data transmission frequencies were determined.

The usability study performed by this research walked thirty users over the basic B-SIPS concepts and functions and then tested their knowledge and the usability of the B-SIPS applications by having the participants perform benchmark tasks. This process uncovered minor adaptations and enhancements in each application that will improve user comprehension and experience. Additionally, it provided the development team with confirmation that their research was well received and viewed as important by members of both the general population and the security world. Future work available in this area includes the completion of all minor adaptations suggested and revealed by the aforementioned usability study, as well as performing additional iterations of the study.

The extension of security and battery lifetime via the alteration of the smart battery polling rate was the next B-SIPS optimization performed. The theory behind this aspect of the research was to alter the polling rate of the smart battery as a means to provide the small mobile device with information more frequently, thus allowing malicious network traffic to be detected and disarmed in a more timely fashion. Both static and dynamic optimizing solutions were devised in this endeavor, each with their own advantages and disadvantages. The static solutions provided have the capacity to be implemented easily in the future with little additional work, but do not increase the security, and thus battery lifetime, of the mobile device nearly as drastically as the dynamic solution does. Conversely, the

dynamic solution allows much more security and power conserving possibilities, but requires drastic revisions to the bus system that currently transmits data between the small mobile device and its smart battery. Both solutions offer capabilities far outweighing the currently implemented smart battery polling rate. Future work in this area is available in exploring the hardware implementation of the theoretical work devised by this research.

A large scale network simulation of the B-SIPS application was completed using the commercial product OPNET. The purpose of this simulation endeavor was to determine throughput degradation rates caused by B-SIPS data transmissions, determine optimal transmission frequencies for various deployment scenarios, and use this data to determine whether or not the deployment of B-SIPS is advisable. As discussed in Chapter 6, this network simulation was designed to reflect the wired-cum-wireless topology structure of the Electrical and Computer Engineering Department at Virginia Tech, with mobile devices running B-SIPS being located in both Durham and Whittemore halls. Simulations showed that external applications experience little to no service degradation unless the network was extremely saturated. In such oversaturated networks, it was determined that the default 1 Hz B-SIPS data transmission period minimized the degradation experienced by both the B-SIPS and external applications. Ultimately, the OPNET simulations provided the necessary confidence to promote the deployment of B-SIPS in large scale network scenarios. Future work in this area is available via an extended exploration of the effects of B-SIPS on TCP-based applications operating in load saturated networks. Additionally, an investigation of simulated attacks in a mobile network that is actively running the B-SIPS application would be useful.

In conclusion, this research helped to aid the security of mobile devices by validating and improving upon the important research endeavor coined B-SIPS. Via optimizations in application usability and provision of deployment setting suggestions, this research intends to promote the use of B-SIPS. This use will hopefully help both casual users, who utilize their small mobile devices for personal recreation, and professionals, who use their devices for crucial work related issues, to experience more stable, secure, and enjoyable use of their hardware. Additionally, this research will help protect the applications and cooperate trade secrets that said users access with their devices. This research also

provides opportunities for hardware manufacturers to increase the effectiveness of the B-SIPS tool by implementing optimized smart battery polling rate mechanisms. B-SIPS is now a more viable, usable, deployable, and secure application suite due to the optimizations performed by this research.

REFERENCES

[1] ACPI, "Advanced configuration and power interface," http://www.acpi.info, 2005.

[2] M. Brownfield, G. Yatharth, et al., "Wireless sensor network denial of sleep attack," in *the Sixth Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop.* West Point, NY, 2005.

[3] T. K. Buennemeyer, M. Gora, et al., "Battery exhaustion attack detection with small handheld mobile computers," *in the IEEE International Conference on Portable Information Devices (Portable '07).* Orlando, FL, 2007.

[4] T. K. Buennemeyer, F. Munshi, et al., "Battery-sensing intrusion protection for wireless handheld computers using a dynamic threshold calculation algorithm for attack detection," *in the 40th Annual Hawaii International Conference on System Sciences (HICSS-40).* Waikoloa, Hawaii, 2007.

[5] T.K. Buennemeyer, T.M. Nelson, L.M. Clagett, J.P. Dunning, R.C. Marchany, and J.G. Tront, "Mobile Device Profiling and Intrusion Detection using Smart Batteries," for the 41st Annual Hawai'i International Conference on System Sciences (HICSS-41), IEEE Computer Society, Waikoloa, HI, 7-10 January, 2008.

[6] T.K. Buennemeyer, T.M. Nelson, M.A. Gora, R.C. Marchany, and J.G. Tront, "Battery Polling and Trace Determination for Bluetooth Attack Detection in Mobile Devices," In the Proceedings of the Eighth Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop, U.S. Military Academy, West Point, NY, 20-22 June, 2007.

[7] T.K. Buennemeyer, T.M. Nelson, R.C. Marchany, and J.G. Tront, "Polling the Smart Battery for Efficiency: Lifetime Optimization in Battery-Sensing Intrusion Protection Systems," In the Proceedings of the IEEE Southeast Conference (SoutheastCon 2007), Richmond, VA, 22-25 March, 2007.

[8] T.K. Buennemeyer, J.G. Tront, R.C. Marchany, P. Schaumont, "Battery-Based Intrusion Detection System", unpublished dissertation, 2007.

[9] Daniel E. Castle, Aaron Darensburg, Barak Griffin, Thadeus Hickman, Stuart P. Warders, and Grant A. Jacoby, "Gibraltar: A Mobile Host-Based Intrusion Protection System," National Conference on Undergraduate Research, pp. 1-8, April 2006.

[10] Epanorama.net, "Serial buses information page," http://www.epanorama.net/links/serialbus.html, 2006.

[11] Clark Gaylord, senior network architect at Virginia Tech.

[12] K. Fall, K. Varadhan, et al., "The ns Manual (formerly *ns* Notes and Documentation)," The VINT Project, A collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC. http://www.isi.edu/nsnam/ns/doc/index.html.

[13] O. Gudmundsson, "RFC 3226 – DNSSEC and IPv6 aware/resolver message size requirements", http://www.faqs.org/rfcs/rfc3226.html, December 2001.

[14] H. Rex Hartson, Deborah Hix, "CS5714 Usability Engineering", used as course notes for Virginia Tech's Usability Engineering class, Fall 2006.

[15] IBM, "Performance Management Guide", http://www.ncsa.uiuc.edu/UserInfo/Resources/Hardware/IBMp690/IBM/usr/share/man/info/en_US/a_doc_lib/aixbman/prftungd/2365c93.htm, 2008.

[16] Institutional Review Board, "Researchers / Forms", 1 April 2008, at 15:10. http://www.irb.vt.edu/

[17] Intel Corporation, "Intel® PXA270 Processor *Electrical, Mechanical, and Thermal Specification,*" http://www.phytec.com/pdf/datasheets/PXA270_DS.pdf, 2005.

[18] G. A. Jacoby and N. J. Davis, "Battery-based intrusion detection," in the IEEE Global Telecommunications Conference (GLOBECOM '04). Dallas, TX, 2004.

[19] J. Kurose and K. Ross, Computer networking: a top-down approach featuring the Internet, 2nd ed.: Addison Wesley, 2002.

[20] T. Martin, M. Hsiao, et al., "Denial-of-service attacks on battery-powered mobile computers," in *the Second IEEE Annual Conference on Pervasive Computing and Communications*. Orlando, FL, 2004.

[21] Microsoft, "Advanced power management v1.2," http://www.microsoft.com/whdc/archive/amp_12.mspx, 2001.

[22] D. C. Nash, T. L. Martin, et al., "Towards an intrusion detection system for battery exhaustion attacks on mobile computing devices," in *the Third IEEE International Conference on Pervasive Computing And Communications Workshops (PerCom '05)*. Kauai Island, HI, 2005.

[23] Nielsen, Jakob, and Landauer, Thomas K.: "A mathematical model of the finding of usability problems," *Proceedings of ACM INTERCHI'93 Conference* (Amsterdam, The Netherlands, 24-29 April 1993), pp. 206-213.

[24] NSA, "An Overview of Wireless Mesh Networks and Current Research", The Next Wave, Fall 2007 issue.

[25] OPNET Technologies, Inc., "Documentation", http://www.opnet.com, 1986-2008.

[26] R. Racic, D. Ma, et al., "Exploiting MMS vulnerabilities to stealthily exhaust mobile phone's battery," in *the 15th Annual USENIX Security Symposium*. Vancouver, BC, 2006.

[27] SANS Institute, "Intrusion Detection Systems; Definition, Need and Challenges," As part of the Information Security Reading Room, 2001.

[28] SBS_Forum, "Smart battery system implementers forum," http://www.sbs-forum.org, 2005.

[29] SMBus, "System management bus," http://www.smbus.org, 2005.

[30] F. Stajano and R. Anderson, "The resurrecting duckling: security issues for ad-hoc wireless networks," in *the 7th International Workshop on Security Protocols*. Cambridge, UK, 1999.

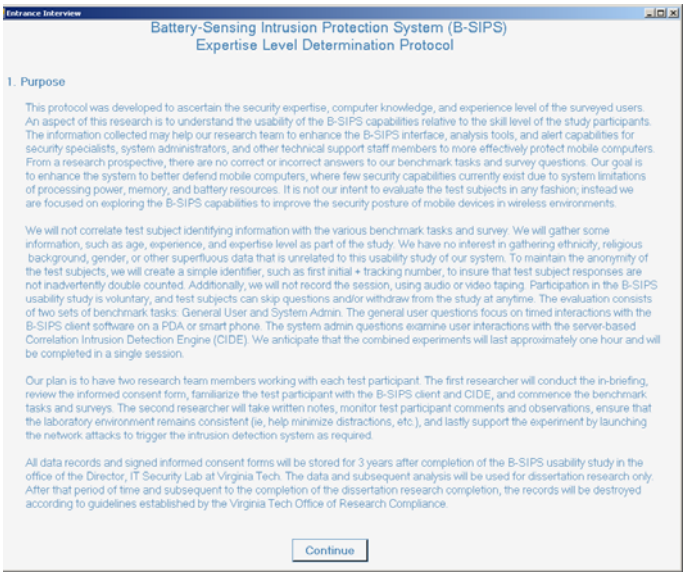
[31] E. Thompson, "Smart batteries to the rescue," http://www.mcc-us.com/SBSRescue.pdf, 2000.

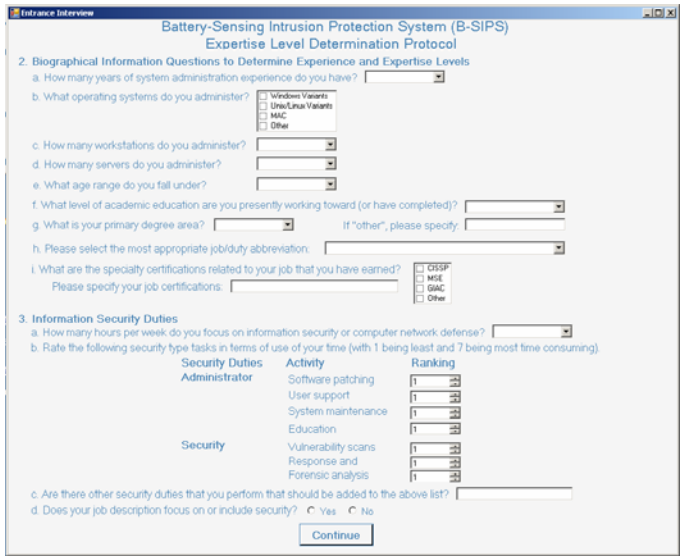
[32] Wikipedia, "Intrusion Detection System," 24 March 2008, at 06:50. http://en.wikipedia.org/wiki/Intrusion_detection_system.

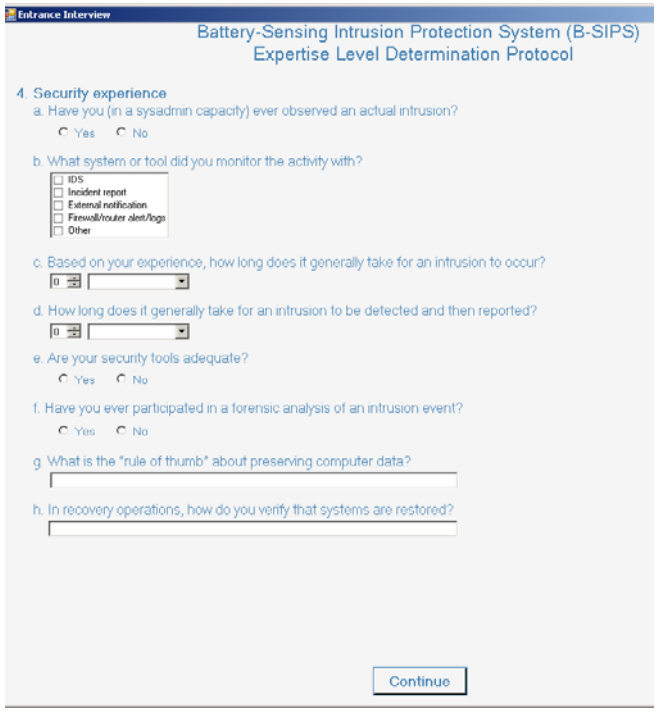
APPENDIX

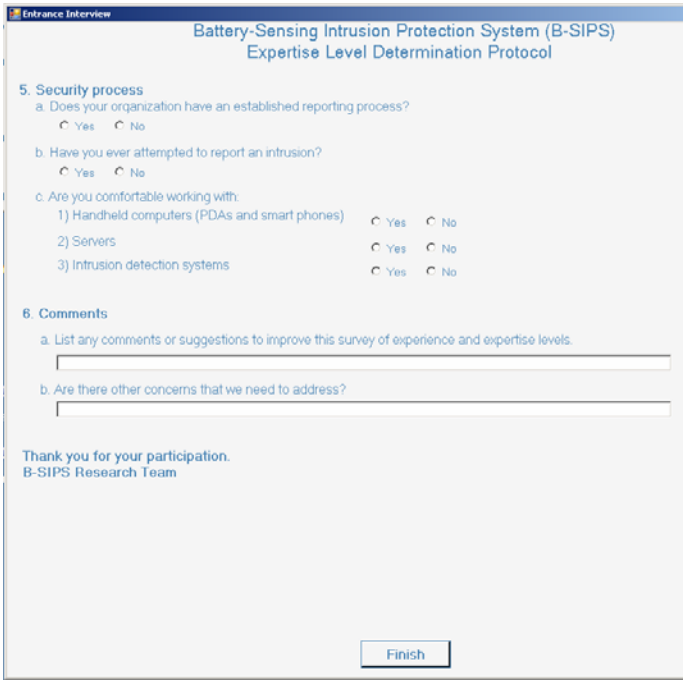
APPENDIX A: APPLICATION USABILITY

APPENDIX A.1: ENTRANCE INTERVIEW

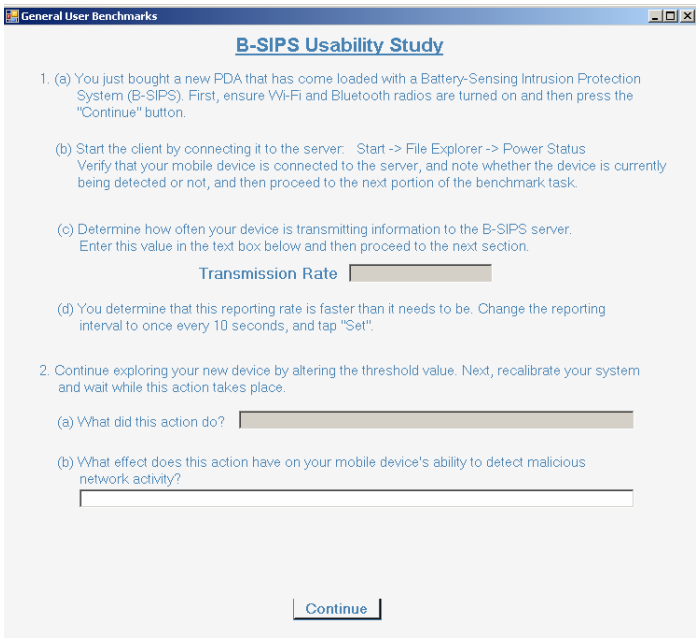


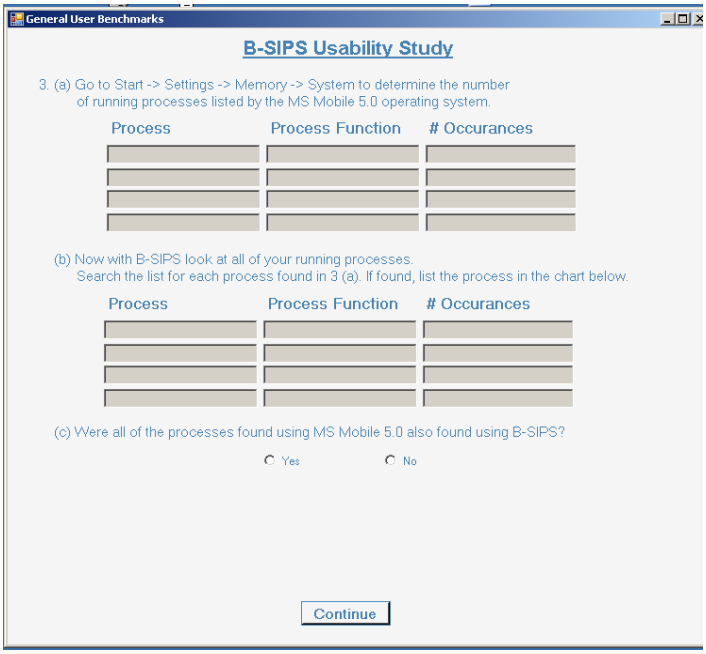


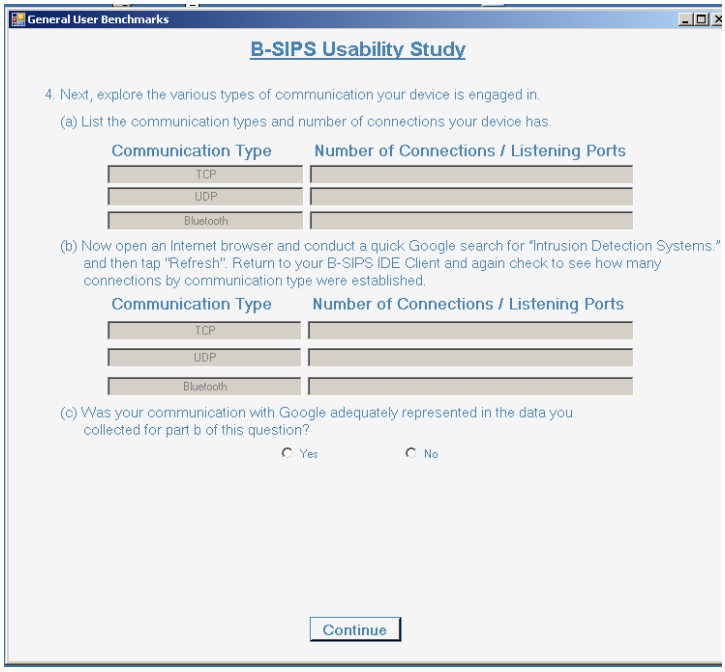

Entrance Interview
Battery-Sensing Intrusion Protection System (B-SIPS)
Expertise Level Determination Protocol
4. Security experience
a. Have you (in a sysadmin capacity) ever observed an actual intrusion?
Yes
No
b. What system or tool did you monitor the activity with?
IDS
Incident report
External notification
Firewall/router alert/logs
Other
c. Based on your experience, how long does it generally take for an intrusion to occur?
d. How long does it generally take for an intrusion to be detected and then reported?
e. Are your security tools adequate?
Yes
No
f. Have you ever participated in a forensic analysis of an intrusion event?
Yes
No
g. What is the "rule of thumb" about preserving computer data?
h. In recovery operations, how do you verify that systems are restored?
Continue

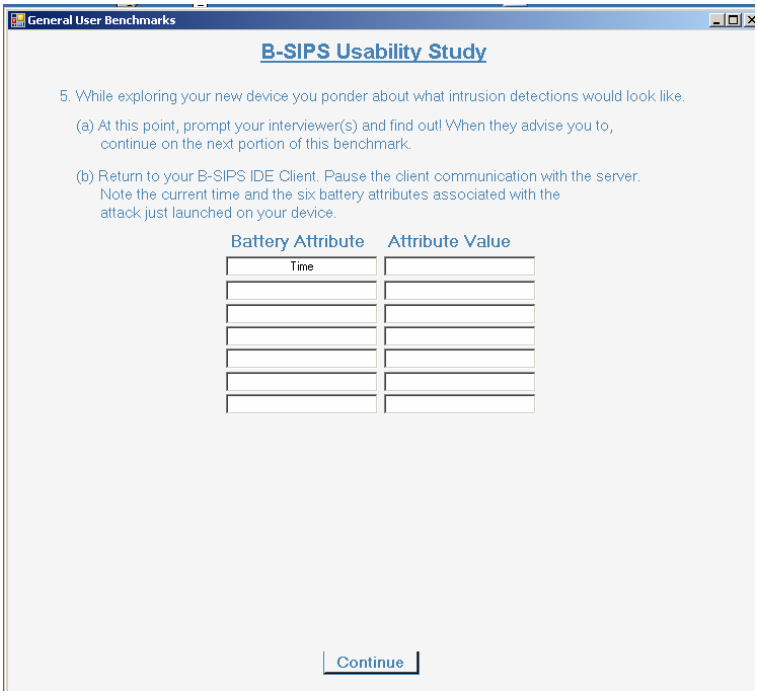

Entrance Interview
Battery-Sensing Intrusion Protection System (B-SIPS)
Expertise Level Determination Protocol
5. Security process
a. Does your organization have an established reporting process?
Yes
No
b. Have you ever attempted to report an intrusion?
Yes
No
c. Are you comfortable working with:
1) Handheld computers (PDAs and smart phones)
Yes
No
2) Servers
Yes
No
3) Intrusion detection systems
Yes
No
6. Comments
a. List any comments or suggestions to improve this survey of experience and expertise levels.
b. Are there other concerns that we need to address?
Thank you for your participation.
B-SIPS Research Team
Finish

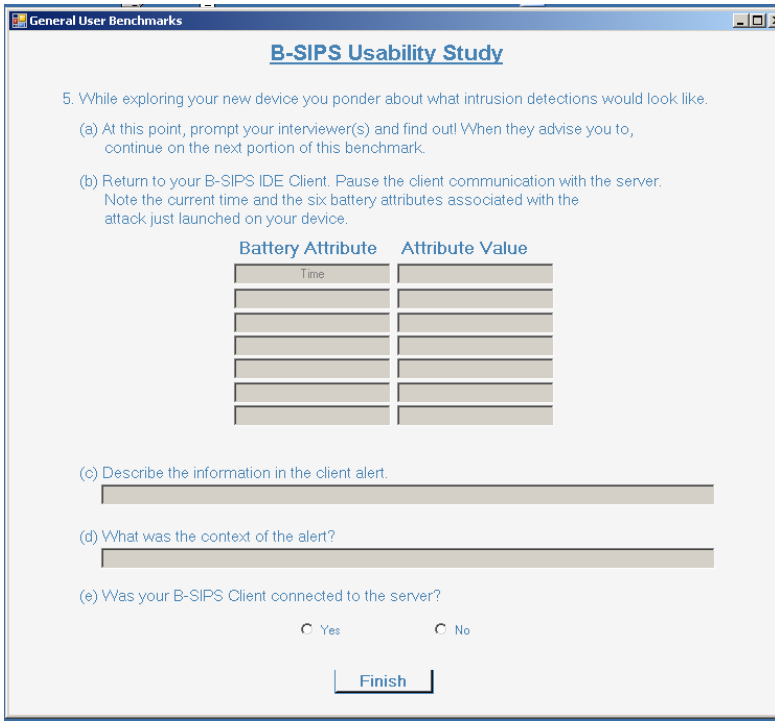
APPENDIX A.2: CLIENT BENCHMARK TASKS



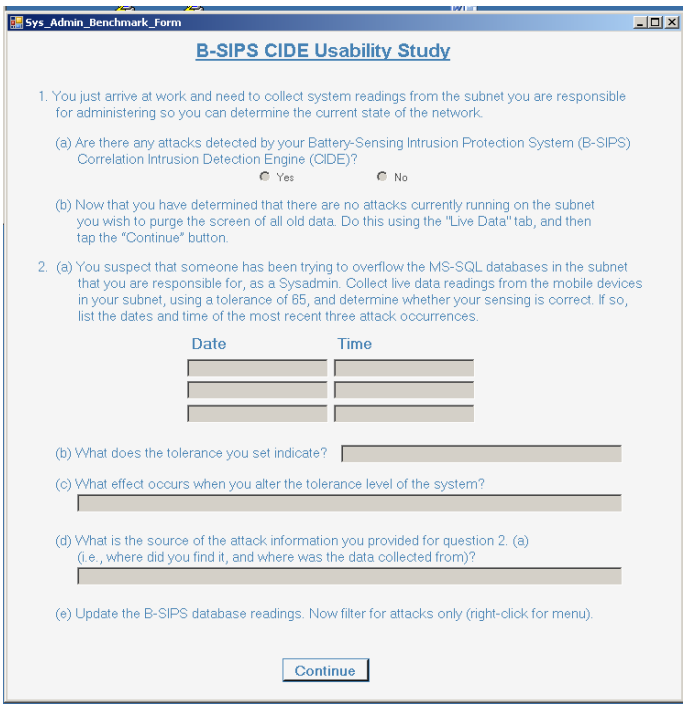


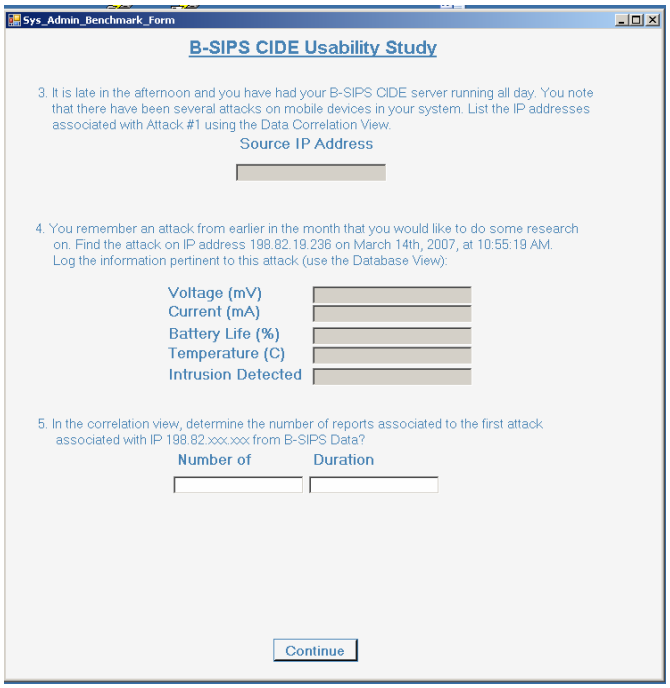

General User Benchmarks
B-SIPS Usability Study
4. Next, explore the various types of communication your device is engaged in.
(a) List the communication types and number of connections your device has.
Communication Type
Number of Connections / Listening Ports
TCP
UDP
Bluetooth
(b) Now open an Internet browser and conduct a quick Google search for "Intrusion Detection Systems." and then tap "Refresh". Return to your B-SIPS IDE Client and again check to see how many connections by communication type were established.
Communication Type
Number of Connections / Listening Ports
TCP
UDP
Bluetooth
(c) Was your communication with Google adequately represented in the data you collected for part b of this question?
Yes
No
Continue

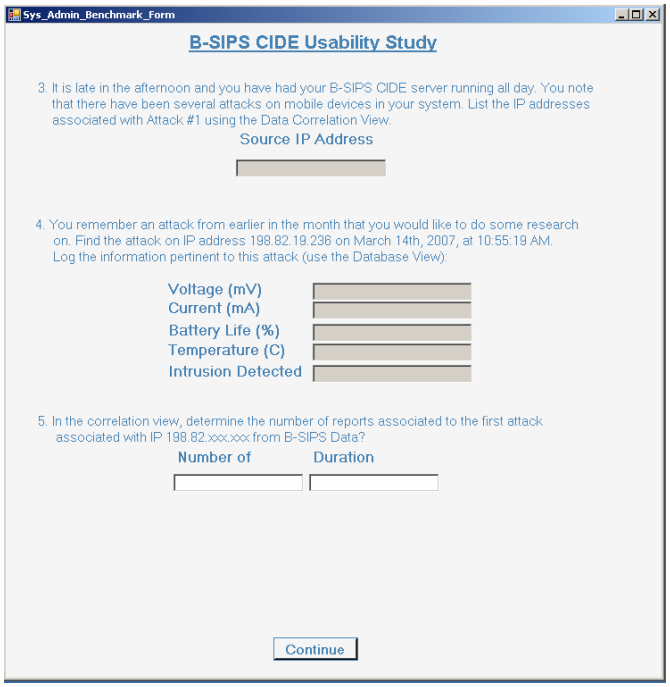

General User Benchmarks
B-SIPS Usability Study
5. While exploring your new device you ponder about what intrusion detections would look like.
(a) At this point, prompt your interviewer(s) and find out! When they advise you to, continue on the next portion of this benchmark.
(b) Return to your B-SIPS IDE Client. Pause the client communication with the server. Note the current time and the six battery attributes associated with the attack just launched on your device.
Battery Attribute
Attribute Value
Time
Continue

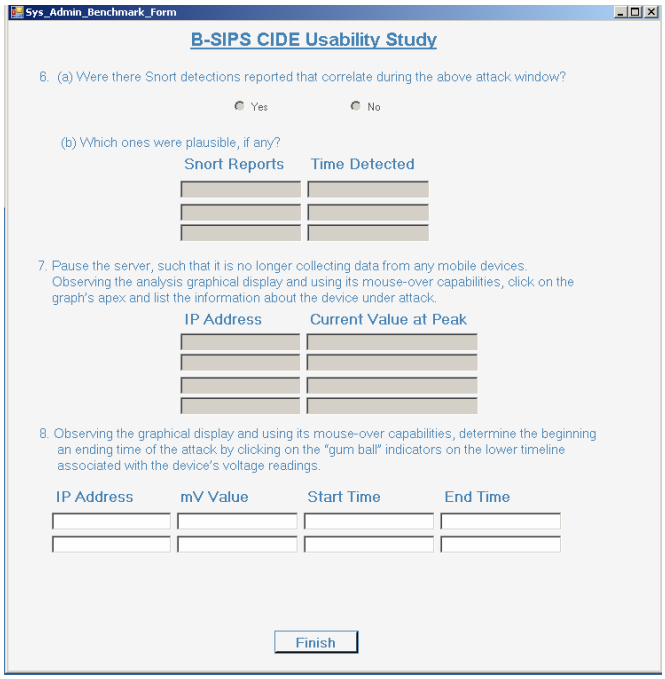

General User Benchmarks
B-SIPS Usability Study
5. While exploring your new device you ponder about what intrusion detections would look like.
(a) At this point, prompt your interviewer(s) and find out! When they advise you to, continue on the next portion of this benchmark.
(b) Return to your B-SIPS IDE Client. Pause the client communication with the server. Note the current time and the six battery attributes associated with the attack just launched on your device.
Battery Attribute
Attribute Value
Time
(c) Describe the information in the client alert.
(d) What was the context of the alert?
(e) Was your B-SIPS Client connected to the server?
Yes
No
Finish

APPENDIX A.3: SYSTEM ADMIN BENCHMARK TASKS

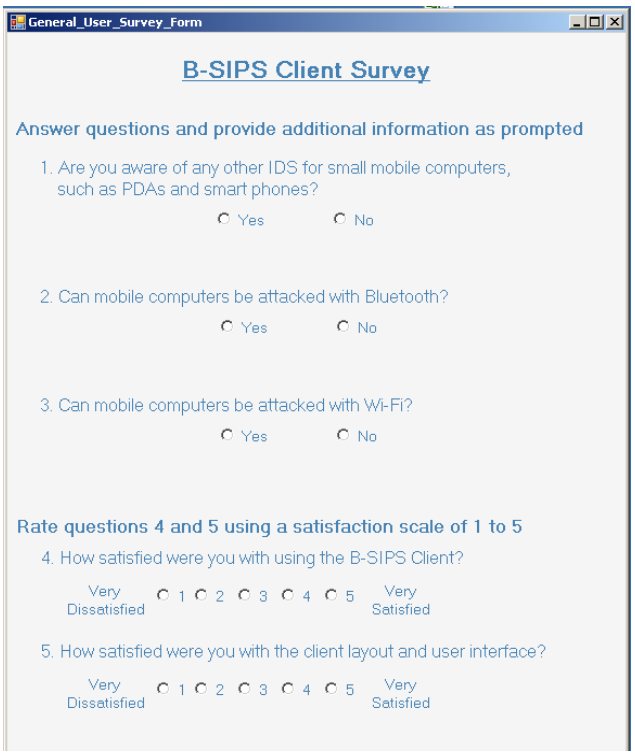


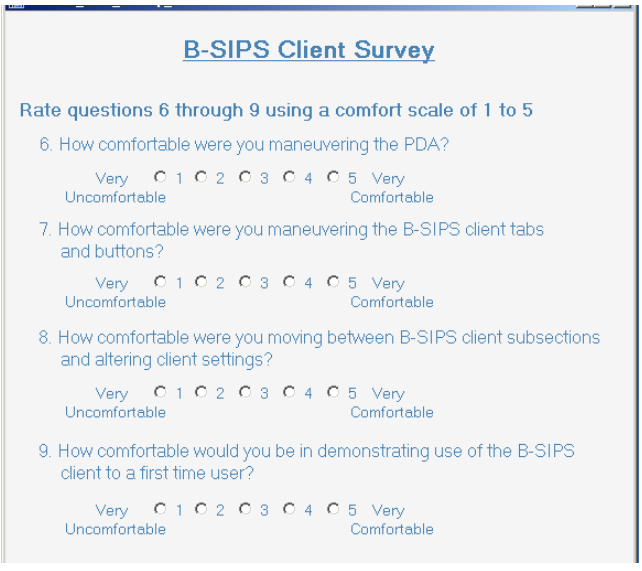


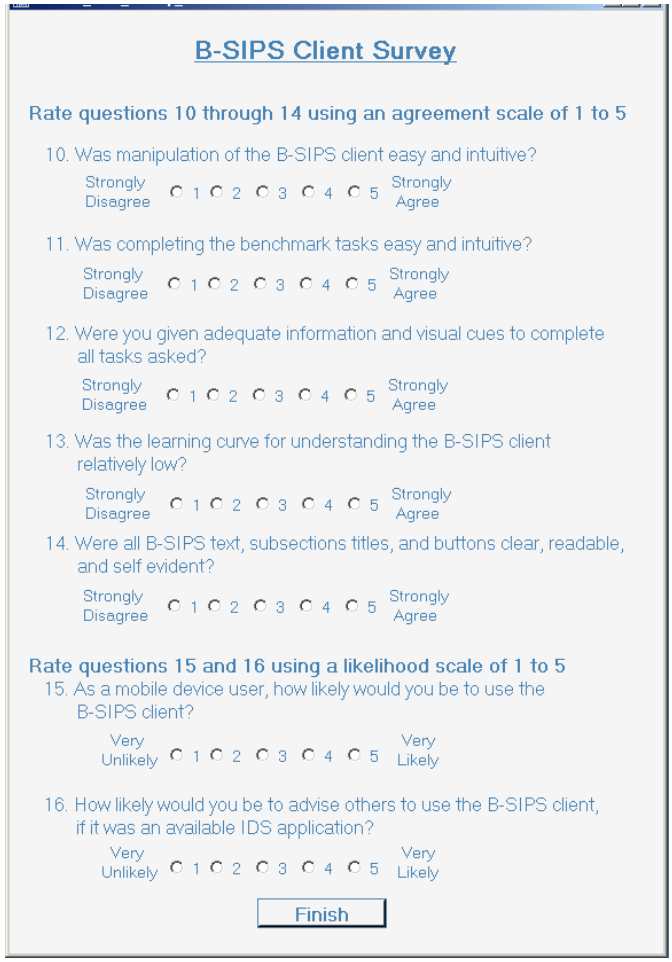

Sys_Admin_Benchmark_Form
B-SIPS CIDE Usability Study
3. It is late in the afternoon and you have had your B-SIPS CIDE server running all day. You note that there have been several attacks on mobile devices in your system. List the IP addresses associated with Attack #1 using the Data Correlation View.
Source IP Address
4. You remember an attack from earlier in the month that you would like to do some research on. Find the attack on IP address 198.82.19.236 on March 14th, 2007, at 10:55:19 AM. Log the information pertinent to this attack (use the Database View):
Voltage (mV)
Current (mA)
Battery Life (%)
Temperature (C)
Intrusion Detected
5. In the correlation view, determine the number of reports associated to the first attack associated with IP 198.82.xxx.xxx from B-SIPS Data?
Number of
Duration
Continue


Sys_Admin_Benchmark_Form
B-SIPS CIDE Usability Study
6. (a) Were there Snort detections reported that correlate during the above attack window?
Yes
No
(b) Which ones were plausible, if any?
Snort Reports
Time Detected
7. Pause the server, such that it is no longer collecting data from any mobile devices. Observing the analysis graphical display and using its mouse-over capabilities, click on the graph's apex and list the information about the device under attack.
IP Address
Current Value at Peak
8. Observing the graphical display and using its mouse-over capabilities, determine the beginning an ending time of the attack by clicking on the "gum ball" indicators on the lower timeline associated with the device's voltage readings.
IP Address
mV Value
Start Time
End Time
Finish

APPENDIX A.4: CLIENT SURVEY







B-SIPS Client Survey

Rate questions 10 through 14 using an agreement scale of 1 to 5

10. Was manipulation of the B-SIPS client easy and intuitive?

Strongly Disagree ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Strongly Agree

11. Was completing the benchmark tasks easy and intuitive?

Strongly Disagree ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Strongly Agree

12. Were you given adequate information and visual cues to complete all tasks asked?

Strongly Disagree ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Strongly Agree

13. Was the learning curve for understanding the B-SIPS client relatively low?

Strongly Disagree ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Strongly Agree

14. Were all B-SIPS text, subsections titles, and buttons clear, readable, and self evident?

Strongly Disagree ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Strongly Agree

Rate questions 15 and 16 using a likelihood scale of 1 to 5

15. As a mobile device user, how likely would you be to use the B-SIPS client?

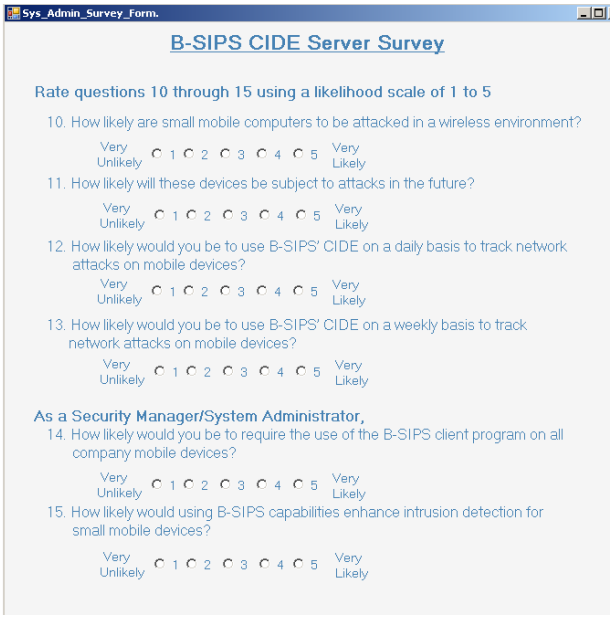
Very Unlikely ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Very Likely

16. How likely would you be to advise others to use the B-SIPS client, if it was an available IDS application?

Very Unlikely ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 Very Likely

Finish

APPENDIX A.5: SYSTEM ADMIN SURVEY



APPENDIX A.6: INFORMED CONSENT FORM

Informed Consent Form for Participant of Investigative Project

Virginia Polytechnic Institute and State University

Title of Project: *Battery-Sensing Intrusion Protection System:* System Interface Evaluation

Principal Investigator: Dr. Joseph G. Tront

I. THE PURPOSE OF THIS RESEARCH/PROJECT

You are invited to participate in a study of Battery-Sensing Intrusion Protection System (B-SIPS). The B-SIPS client is a network security tool for mobile devices that monitors battery usage to detect malicious network activity. The B-SIPS server collects pertinent data, transmitted by the mobile devices running the B-SIPS client, and correlated network activity for use by System Administrators. This study involves experimentation for the purpose of evaluating and improving the B-SIPS user interface.

II. PROCEDURES

You will be asked to perform a set of tasks using B-SIPS. These tasks consist of starting and manipulating application settings on the mobile device, as well as monitoring results on a B-SIPS correlating server console. Your role in these tests is that of evaluator of the software. We are not evaluating you or your performance in any way; you are helping us to evaluate our system. All information that you help us attain will remain anonymous. Your actions will be noted and you will be asked to describe verbally your learning process. You may be asked questions during and after the evaluation, in order to clarify our understanding of your evaluation. You may also be asked to fill out a questionnaire relating to your usage of the system.

The session will last about 1 hour. There are no risks to you. The tasks are not very tiring, but you are welcome to take rest breaks as needed. If you prefer, the session may be divided into two shorter sessions. You may also terminate your participation at any time, for any reason, no questions asked.

III. RISKS

There are no known risks to the subjects of this study.

IV. BENEFITS OF THIS PROJECT

Your participation in this project will provide information that may be used to improve the usability of B-SIPS. No guarantee of benefits has been made to encourage you to participate. You may receive a synopsis summarizing this research when completed. Please leave a self-addressed envelope with the experimenter and a copy of the results will be sent to you.

You are requested to refrain from discussing the evaluation with other people who might be in the candidate pool from which other participants might be drawn.

V. EXTENT OF ANONYMITY AND CONFIDENTIALITY

The results of this study will be kept strictly confidential. No one outside the project team will be able to connect any data with your name. The information you provide will have your name removed and only a subject number will identify you during analyses and any written reports of the research. Neither a video nor audio recording will be made of the experimental session with you.

VI. COMPENSATION

Your participation is voluntary and unpaid. Students enrolled in the Network Security (ECE 4560) course will receive homework credit for their participation; this credit amounts to 2% of their final grade.

VII. FREEDOM TO WITHDRAW

You are free to withdraw from this study at any time for any reason.

VIII. APPROVAL OF RESEARCH

This research has been approved, as required, by the Institutional Review Board for projects involving human subjects at Virginia Polytechnic Institute and State University.

IX. SUBJECT'S RESPONSIBILITIES AND PERMISSION

I voluntarily agree to participate in this study, and I know of no reason I cannot participate. I have read and understand the informed consent and conditions of this project. I have had all my questions answered. I hereby acknowledge the above and give my voluntary consent for participation in this project. If I participate, I may withdraw at any time without penalty. I agree to abide by the rules of this project

_____ _____

Signature Date

_____ _____

Name (please print) Contact: phone or address or

_____

email address (OPTIONAL)

X. EXPERIMENTERS' RESPONSIBILITIES

As the person(s) responsible for conducting the empirical study described in this form, we agree that we are responsible for protecting participant/subject rights and meeting the conditions described here.

Dr. Joseph Tront		
Name (please print)	Signature	Date
Timothy Buennemeyer		
Name (please print)	Signature	Date
Theresa Nelson		
Name (please print)	Signature	Date
Lee Clagett		
Name (please print)	Signature	Date
John Paul Dunning		
Name (please print)	Signature	Date

To the participant/subject: Should you have any questions about this research or its conduct, you may contact:

Investigators:

Timothy Buennemeyer Phone (540) 230-5203
Graduate student, Electrical & Computer Engineering Department
email: timb@vt.edu

Theresa Nelson Phone (540) 250-0971
Graduate student, Electrical & Computer Engineering Department
email: tnelson@vt.edu

Lee Clagett Phone (410) 208-9920
Undergraduate student, Electrical & Computer Engineering Department
email: lclagett@vt.edu

John Paul Dunning Phone (336) 210-4070
Undergraduate student, Electrical & Computer Engineering Department
email: jpvt40@vt.edu

Advisor:

Dr. Joseph G. Tront Phone (540) 231-4857
Professor, Electrical & Computer Engineering Department
email: jgtront@vt.edu

Review Board:

David M. Moore, Chair, IRB
Research Compliance Office
CVM Phase II (0442)
(540) 231-4991
Virginia Tech

APPENDIX B: SMART BATTERY POLLING RATE

```
function poll_rate = opt_poll(),

format short g;
min_poll_time = 0.00879;
idle = 0.07222;
active = 0.25694;
vcc_battery = 3;
battery_lifetime = 1100.00000;
j=0;

for i = 1/100:1/100:600,
j=j+1;
poll_time(j) = i;
polls_per_min(j) = 60/poll_time(j);
time_spent_polling(j) = polls_per_min(j) * min_poll_time;
mW_per_min(j) = (time_spent_polling(j) * active) + ((60-time_spent_polling(j)) *
idle);
mA_per_min(j) = mW_per_min(j) / vcc_battery;
lifetime_min(j) = battery_lifetime / mA_per_min(j);
lifetime_hrs(j) = lifetime_min(j) / 60;
end

plot(poll_time, lifetime_hrs, 'k');
hold on;
attacks = [0, 1, 10, 100, 1000, 10000, 20000, 30000, 40000, 50000, 60000, 70000,
80000, 90000, 100000];
color = ['b', 'r', 'b', 'r', 'b'];

for a = 1:1:15,
j=0;
threshold_y(a) = 0;
threshold_x(a) = 0;
for i = 1/100:1/100:600,
j=j+1;
poll_time(j) = i;
attack_time(j) = (poll_time(j)+min_poll_time) * attacks(a);
attack_mW = attack_time(j) * active;
attack_mA = attack_mW / vcc_battery;
reg_mA = battery_lifetime - attack_mA;
reg_mW = reg_mA * vcc_battery;
reg_min = reg_mW / mW_per_min(j);
attack_lifetime_min(j) = (attack_time(j) / 60) + reg_min;
if attack_lifetime_min(j) < 0
attack_lifetime_min(j) = 0;
end
attack_lifetime_hrs(j) = attack_lifetime_min(j) / 60;

% Look at tolerance between the two
if attack_lifetime_hrs(j) > threshold_y(a)
threshold_x(a) = i;
threshold_y(a) = attack_lifetime_hrs(j);
end
end
```

```
%poll_rate(a, 1) = attacks(a);
poll_rate(a, 1) = threshold_x(a);

if a < 6
plot(poll_time, attack_lifetime_hrs, color(a));
plot(threshold_x(a), threshold_y(a), 'ko' );
end
end

% Format the table
xlabel('Frequency of Smart Battery Poll (sec)','FontSize',12, 'FontWeight','bold');
ylabel('Lifetime (hours)','FontSize',12, 'FontWeight','bold');
title('The Effect of Battery Polling on PDA Lifetime', 'FontSize',14,
'FontWeight','bold');
set(gcf,'Color','white');
%legend('No Attacks', '1 Attack', 'Optimum Rate', '10 Attacks', 'Optimum Rate', '100
Attacks', 'Optimum Rate', '1,000 Attacks', 'Optimum Rate', '10,000 Attacks', 'Optimum
Rate');
```

Figure 28. MATLAB code for optimizing a static polling rate given a known network density

```
function lifetime_hrs = dynamic(poll_type, dyn_poll_time_min, dyn_poll_time_max),

format short g;
attacks = [0, 1, 10, 100, 1000, 10000, 20000, 30000, 40000, 50000, 60000, 70000,
80000, 90000, 100000];
inform_poll_time = 0.00048;
static_min_poll_time = 0.00879;
min_poll_time = inform_poll_time + static_min_poll_time;
idle = 0.07222;
active = 0.25694;
vcc_battery = 3;
battery_lifetime = 1100.00000;
poll_time = dyn_poll_time_max;


if poll_type == 1
% Static Polling
for a = 1:1:15,
attack_count = 0;
attack_time = 0;
attack_mW = 0;
attack_count = 0;
reg_time = 0;
reg_mW = 0;
reg_mA = 0;

leftover_mW = battery_lifetime * vcc_battery;
actual_reg_time = 0;

while leftover_mW > 0
if (attack_count < attacks(a))
```

```
attack_time = attack_time + poll_time;
attack_mW = poll_time * active;
attack_count = attack_count + 1;
leftover_mW = leftover_mW - attack_mW;
else
reg_time = poll_time + static_min_poll_time;
reg_mW = (poll_time * idle) + (static_min_poll_time * active);

if leftover_mW > reg_mW
actual_reg_time = actual_reg_time + reg_time;
leftover_mW = leftover_mW - reg_mW;
else
actual_reg_time = actual_reg_time + (leftover_mW / reg_mW * reg_time);
leftover_mW = 0;
end
end
end

attacks_launched = attack_count;
lifetime_sec = (attack_time / 60) + actual_reg_time;
lifetime_min = lifetime_sec / 60;
lifetime_hrs(a,1) = lifetime_min / 60;
end
end

if poll_type == 2
% Clustered Attacks
for a = 1:1:15,
attack_count = 0;
attack_time = 0;
attack_mW = 0;
attack_count = 0;
reg_time = 0;
reg_mW = 0;
reg_mA = 0;

leftover_mW = battery_lifetime * vcc_battery;
actual_reg_time = 0;

while leftover_mW > 0
if (attack_count < attacks(a))
attack_time = attack_time + poll_time;
attack_mW = poll_time * active;
attack_count = attack_count + 1;
poll_time = dyn_poll_time_min;
leftover_mW = leftover_mW - attack_mW;
else
reg_time = poll_time + min_poll_time;
reg_mW = (poll_time * idle) + (min_poll_time * active);

if leftover_mW > reg_mW
actual_reg_time = actual_reg_time + reg_time;
leftover_mW = leftover_mW - reg_mW;
else
actual_reg_time = actual_reg_time + (leftover_mW / reg_mW * reg_time);
leftover_mW = 0;
end
```

```
if poll_time < dyn_poll_time_max
poll_time = poll_time * 2;
end
if poll_time > dyn_poll_time_max
poll_time = dyn_poll_time_max;
end
end
end

attacks_launched = attack_count;
lifetime_sec = (attack_time / 60) + actual_reg_time;
lifetime_min = lifetime_sec / 60;
lifetime_hrs(a,1) = lifetime_min / 60;
end
end

if poll_type == 3
% Distributed Attacks
for a = 1:1:15,
attack_count = 0;
attack_time = 0;
attack_mW = 0;
attack_count = 0;
reg_time = 0;
reg_mW = 0;
reg_mA = 0;

leftover_mW = battery_lifetime * vcc_battery;
actual_reg_time = 0;

while leftover_mW > 0
%if (poll_time > dyn_poll_time_min & attack_count < attacks(a))
if (poll_time == dyn_poll_time_max & attack_count < attacks(a))
attack_time = attack_time + poll_time;
attack_mW = poll_time * active;
attack_count = attack_count + 1;
poll_time = dyn_poll_time_min;
leftover_mW = leftover_mW - attack_mW;
else
reg_time = poll_time + min_poll_time;
reg_mW = (poll_time * idle) + (min_poll_time * active);

if leftover_mW > reg_mW
actual_reg_time = actual_reg_time + reg_time;
leftover_mW = leftover_mW - reg_mW;
else
actual_reg_time = actual_reg_time + (leftover_mW / reg_mW * reg_time);
leftover_mW = 0;
end
if poll_time < dyn_poll_time_max
poll_time = poll_time * 2;
end
if poll_time > dyn_poll_time_max
poll_time = dyn_poll_time_max;
end
end
end
```

```
attacks_launched = attack_count;
lifetime_sec = (attack_time / 60) + actual_reg_time;
lifetime_min = lifetime_sec / 60;
lifetime_hrs(a,1) = lifetime_min / 60;
end
end
```

Figure 29. MATLAB code for optimizing a polling rate over a variety of network densities