Proceedings of the 2007 IEEE
Workshop on Information Assurance
United States Military Academy, West Point, NY 20-22 June 2007

Battery Polling and Trace Determination for Bluetooth Attack Detection in Mobile Devices

Timothy K. Buennemeyer, *Student Member, IEEE*, Theresa M. Nelson, *Student Member, IEEE*, Michael A. Gora, Randy C. Marchany, *Member, IEEE*, and Joseph G. Tront, *Senior Member, IEEE*

***Abstract*—This paper introduces a supporting model for a unique Battery-Sensing Intrusion Protection System (B-SIPS) for mobile computers, which alerts when power changes are detected on small wireless devices. An analytical model is employed to examine the smart battery characteristics to support the theoretical intrusion detection limits and capabilities of B-SIPS. This research explores the modification of the smart battery polling rates in conjunction with the variance of malicious network activity. Using the results from a previous study of optimized static polling rates to create minimum and maximum thresholds, a dynamic polling rate algorithm was devised. This algorithm allowed the smart battery to gauge the network's illicit attack density and adjust its polling rate to efficiently detect attacks, while conserving battery charge life. Lastly, a trace signature methodology is presented that characterizes unique activity for IEEE 802.15.1 (Bluetooth) attack identification.**

***Index Terms*—Battery, Intrusion Detection, Wireless Security**

I. Introduction

THE primary challenges in developing defensive applications such as intrusion detection systems (IDSs) for small, wireless computers are limited processing capability, memory, and battery resources. Traditionally, network and host-based IDSs employ rules to detect known malicious activity. Anomaly detection systems (ADSs) use statistical methods to establish a system profile and then trigger alerts when that normal profile is violated. This research initiative is developing a battery-based detection system that employs mobile devices as sensors that use an instantaneous current-based threshold algorithm to indicate anomalous activity.

An indicator that a rogue process is being run on a device without the knowledge of the user is an unexplained increase in the instantaneous current drawn from a device's battery. This could indicate anomalous activity such as a worm spread, virus infection, network probing, flooding, or denial of service (DoS) attack. All of these malicious activities can cause the battery current to rise such that a well-designed system could detect the illicit activity. The *Battery-Sensing Intrusion Protection System* (B-SIPS) detection capability provides security administrators (SAs) in a network environment with a complementary IDS tool. This nontraditional method can detect anomalous battery exhaustion, IEEE 802.15.1 (Bluetooth) and IEEE 802.11 (Wi-Fi) attack activity that standard IDSs are incapable of detecting [1].

This research further examines various means to refine the B-SIPS detection capabilities. Smart battery polling rates, system management bus speeds, and attack execution times can be used to improve the theoretical accuracy of battery-based anomaly detection. A methodology is then presented to identify an attack, using data sampled by an oscilloscope that is filtered to characterize a unique trace signature.

The rest of this paper is structured as follows. Section II presents related work. Section III describes the ongoing technology convergence trend in mobile devices. Section IV presents B-SIPS' implemented capabilities. Section V discusses the smart battery polling model's design and outlines previous work on testing and optimizing static polling rate solutions. Section VI utilizes the static polling results in formulating a dynamic solution and evaluating it in respect to both the currently implemented polling rate and the optimized static polling rates. Section VII introduces the attack trace determination method. Section VIII presents crafted Bluetooth attacks and traces. Lastly, Section IX provides a conclusion.

II. Background and Related Work

Battery power is an important resource in the wireless domain, especially for small mobile devices. This presents designers with the perplexing problem of choosing more security at the expense of greater power usage and potentially less service availability. Establishing secure communication channels through proper authentication could increase service accessibility from a user's perspective, but it may further increase the device's computational and transmission requirements, leading to faster battery drain.

The Advanced Power Management (APM) specification is an application programming interface which allowed computer and Basic Input Output System (BIOS) manufacturers to include power management in their BIOS and operating systems (OSs), thus reducing energy consumption [2]. Subsequently, the Advanced Configuration and Power Interface (ACPI) established an industry-standard for interfaces to OS directed power management on laptops, desktops, and servers [3]. The Smart Battery System Implementers Forum offered an open systems communication standard for industry-wide adoption that described data sharing between batteries and the devices they powered [4]. Their Smart Battery Data (SBData) specification was used to

T. Buennemeyer is a Ph.D. candidate, T. Nelson is a graduate student, M. Gora is a senior, R. Marchany is the Director of the IT Security Lab, and J. Tront is a Professor of Electrical and Computer Engineering at Virginia Tech. E-mail: {timb, tnelson, gora, marchany, jgtront}@vt.edu.

monitor rechargeable battery packs and to report information to the System Management Bus (SMBus) [5] [6].

Stajano et al. [7] suggested the idea of energy depletion attacks in 1999, which they described as *sleep deprivation torture*. An emerging class of attacks, battery exhaustion and denial of sleep attacks represent malicious situations whereby the device's battery has been unknowingly discharged, and thus the user is deprived access to information [8]. These attacks exploit the power management system by inhibiting the device's ability to shift into reduced power states.

Martin et al. [8] subdivided sleep deprivation attacks against laptop computers. *Service-requesting* attacks try to connect to the mobile device repeatedly with power draining service requests. *Benign* attacks attempt to start a power demanding process or component operation on the host to drain its battery. *Malignant* attacks infiltrate the host and alter programs to devour more battery resources than are typically required.

Racic et al. [9] demonstrated successful battery exhaustion attacks that transited commercial cellular phone networks to exploit vulnerabilities in an insecure multimedia messaging service, context retention in the packet data protocol, and the paging channel. These attacks drained the device's battery, rendering it useless in a short period of time by keeping it in a busy state. Most concerning is the fact that the cellular phone user and network administrator were unaware that the attack was ongoing. An attack of this nature will use more device power, and thus demonstrates the potential effectiveness of an integrated battery-sensing IDS [10].

Nash et al. [11] developed a battery constraints-based IDS for laptop computers aimed toward defending the system against various classes of battery exhaustion attacks. They leveraged the laptop's robust computational power to estimate power consumption of the overall system and then adapted this concept on a per-process basis as a method for indicating possible intrusions and rogue applications.

For mobile handheld devices, Jacoby [12] developed a host-centric *Battery-Based Intrusion Detection* solution. This system was comprised of three distinctive IDS applications. For low power devices, the *Host Intrusion Detection Engine* was a rules-based program tuned to determine battery behavior abnormalities in the busy and idle states using static threshold levels. A complementary *Source Port Intrusion Engine* was employed to capture network packet information during a suspected attack. For robust devices, the *Host Analysis Signature Trace Engin*e was used to capture and correlate spectrum signature patterns using periodogram analysis to determine the dominant frequency and magnitude (*x,y*) pairs. To our knowledge, this system presented the first feasible battery-based IDS solution for handheld devices.

B-SIPS research is developing an innovative battery power constraint-based model and system to help defend small mobile computers and smart phones. Interoperability and low power design were inspired by the demand to significantly increase battery life and thus the usefulness of small mobile hosts. Battery constraint-based intrusion detection and this B-SIPS research endeavor would not be feasible without these technological advances in ACPI and smart batteries.

III. Technology Convergence

In July 2006, *In-Stat* research found that technology convergence was occurring with wireless devices. Although most mobile computer users desire to have a converged technology device, the survey indicated that a significant majority of users carry multiple portable devices with redundant applications and capabilities. *In-Stat* forecasted that the trend to replace laptops and PDAs with smart phones is in the early stages, but that other business forces will drive the adoption of smart phones, such as usefulness, business applications and support for the converged applications [13].

Technology convergence occurs when modern technologies that are capable of performing similar tasks merge. Examples of convergence are synergistic combinations of voice, data, and video into the wireless networking environment [14]. In the past, these capabilities were separate but today provide robust possibilities for sharing resources and interacting with others towards achieving new communication efficiencies. Today, most individuals have access to ubiquitous information because they can communicate through multiple means via mobile phones, communication enhanced Personal Digital Assistants (PDAs), and laptop computers. Often these capabilities are hosted on the same device and alternatively communicated by cellular, Wi-Fi, or Bluetooth.

PDAs were originally designed as personal organizers but became much more versatile over the years. Handheld mobile computing devices are typically used for simple calculating, clock and calendar, Internet access, email, and video applications [14]. A smart phone is any electronic handheld device that integrates the functionality of a mobile phone, PDA, or other information appliance. Typically, telephone communication functions are added to an existing PDA design. The lines are blurred on the direction from which the merger stems, but smart phones are an excellent example of technology convergence. A feature that distinguishes a smart phone from traditional cellular telephone technology is that third party software applications can be installed on the device [14]. These wireless communication and device categories are at the forefront of technology convergence and are being examined in ongoing B-SIPS research on PDAs and smart phones.

IV. B-SIPS Capabilities

B-SIPS provides threshold monitoring and alert notification as a host application, which triggers during detected power changes on small wireless devices. These hosts are employed as sensors in a wireless network and form the basis of the *Canary-Net* IDS [15]. This detection capability is scalable and designed to complement existing commercial and open system network IDSs. B-SIPS correlates device power consumption with Bluetooth and Wi-Fi communication activity. Irregular and attack activity is detected and reported to the server for comparisons against attack trace signatures.

The system was developed in Microsoft C# in the .NET Compact Embedded (CE) environment [16]. The client code was ported to run within Windows CE for Mobile 5.0, and the B-SIPS suite of tools is produced for Dell Axim X51v and Hewlett-Packard iPAQ hx2795 PDAs. The detection tools were employed on Cingular 8125, Verizon XV6700, Palm Treo 700w and Samsung SCH-i730 smart phones operating with the Mobile 5.0 Phone version as well.

B-SIPS detection capability focuses on small mobile hosts that are Bluetooth and Wi-Fi enabled, so conservation of power is of paramount consideration in determining what information is captured, where the information is stored, when the attack signatures are transmitted, and how intrusion correlation is conducted. B-SIPS alert notification is done on the client device for the user and across the network by a server for the SA. Certain power-depleting attacks such as floods, buffer overflows, and various DoS attacks can be profiled by their pulsing pattern or continuous high drain characteristics, while other attacks merely create temporary spikes in power usage and are much more difficult to pattern. B-SIPS is an ADS and IDS hybrid because it attempts to match power traces of some known attacks and correlate the attack with other network IDSs.

B-SIPS uses battery constraints and current thresholds to trigger device alerts in idle and busy states. The potential for false positives and false negatives is of great concern. B-SIPS strives to minimize both through dynamic threshold tuning. Also, the system attempts to correlate alerts with packet header information for forensic analysis. B-SIPS detects anomalous activity that exceeds the system's dynamic threshold value. The *Dynamic Threshold Calculation* (DTC) algorithm iteratively considers known device processes, backlighting, and system states [1]. Although false positives are a possibility with any detection system, B-SIPS is less prone to false positive alerts because the DTC considers normal device power draining activities and then only triggers an alert when the threshold is exceeded by the device's response to anomalous activity.

B-SIPS calculates the DTC value for comparison with the battery's instantaneous current reading. However, the smart battery only provides the instantaneous current reading once per second, at best, due to limitations in the smart battery chipset. When a threshold breach occurs, B-SIPS transmits reports to a server running the *Correlation Intrusion Detection Engine* (CIDE). The reporting continues while the DTC value is exceeded. Although rapid reporting has a strong potential benefit for early detection and corrective actions by the SA, there is a clear tradeoff in that the client device will expend additional energy to transmit a potentially high volume of reports that could reduce the useful battery life of the device. CIDE compares the transmitted reports against a trace signature database, and the increased PDA energy drain is graphically represented to alert the SA in near real-time.

V. Smart Battery Optimization using Static Polling

When a mobile device is kept in a high activity state for extended periods of time, the battery power is depleted faster than normal, decreasing its expected charge life. This research seeks to protect the device's battery charge life by detecting anomalous battery draining activities. The research goal is to optimize the polling period mobile devices use to determine malicious energy consumption. An assumption was made that once an attack is detected it is immediately eliminated, thus affording each attack with a maximum duration of one smart battery polling time interval in the model.

Precisely timed attacks have the potential to defeat the B-SIPS client detection capabilities. If the attacker knows the precise timing of the polling rate of the battery's chipset, then the attacker could attempt to craft intrusion packets to arrive within those limited time windows between the battery's polling intervals, as shown in Fig. 1. The smart battery specification recommends current sampling at least once every five seconds [4]. So packet crafting is a possibility, although remote. B-SIPS' answer to this issue is that the attacker will most likely be unable to manipulate both his attack's timing and the energy usage of the targeted device simultaneously. Since the attack is transiting a wireless environment, the timing would be even more difficult, if not impossible to control. Alternatively, if the smart battery could be designed to randomly sample its instantaneous current within that time interval and still provide comparable performance and diagnostic readings, then this precise timing attack manipulation would be exceedingly difficult to execute [17].

This leads to a noted limitation that the smart battery provides its diagnostic readings, at best, only once per second. At present, original equipment manufacturers (OEMs) have built this generation of smart batteries to provide a limited set of information to the OS for managing the device's power usage and recharging the battery. In the future, if OEMs could improve the smart battery's chipset to poll at a faster rate to accommodate the needs of battery-based IDSs, the timing attack window concern would be mitigated. This would provide the added benefit of potentially helping B-SIPS detect more attacks. The idea hinges on the fact that certain attacks could occur at rates exceeding the battery's sampling speed, so those attacks could be missed. This research is being conducted to determine the typical speed of attack executions with regard to current device processing rates and bus speeds. Although B-SIPS cannot solve this issue, this research may suggest the proper sampling speed for next generation smart batteries to further enhance the detection system's capabilities.

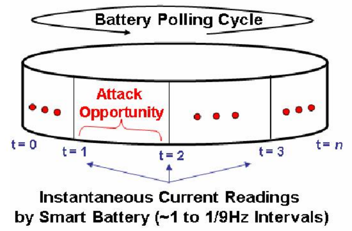
Battery Polling Cycle
Attack
Opportunity
t = 0
t = 1
t = 2
t = 3
t = n
Instantaneous Current Readings
by Smart Battery (~1 to 1/9Hz Intervals)

Fig. 1. Battery polling cycle timing attack window.

With an aim towards protecting a device's battery life, detection of anomalous battery draining activities has become the research focus for optimizing the smart battery's polling mechanism to enhance B-SIPS' attack detection capability. This optimized polling mechanism was designed with the mindset that, as the interval between battery information transmissions is decreased, the time that an attack goes undetected is reduced.

Using the Dell Axim X51 PDA as a testing platform for analysis, hardware factors included smart battery mW hours and voltage settings, processor idle and active mW rates, and required data transfer. Once these factors were set corresponding to the Dell Axim X51 device, lifetime was calculated. These calculations were dependent on the battery polling rate and the number of attacks the device was subjected to, coined as its *network attack density*.

A *MATLAB* model was constructed which found the optimum polling rate for predetermined network attack frequency by calculating the battery charge life with 60,000 different battery polling periods, ranging from 0.00167Hz (once every 600 seconds) to 100Hz (once every 0.01 seconds). This procedure was repeated for attack densities of 0, 1, 10, 100, 1,000, and 10,000 – 100,000, in increments of 10,000. A graphical depiction of the results is shown in Fig. 2.

Once the optimum static polling rates were determined, a second function used them to calculate the device charge life for each of the 15 network attack densities. Data obtained from this function showed that none of the calculated polling rates performed acceptably under all tested conditions. Table I illustrates the battery charge life results of two mutually exclusive lifetime enhancing techniques: overhead reduction in low network attack densities by refraining from polling more often than necessary, and quickly detecting and disabling malicious traffic in high network densities by polling the smart battery more frequently. The mutual exclusion of these life increasing techniques leads to the conclusion that static polling solutions are not suitable for maximizing device lifetime [17].

VI. Educating Smart Batteries via Dynamic Polling

This section outlines the advantages and disadvantages that dynamic approaches bring to power draining network attack detection systems, as well as introducing a dynamic polling rate algorithm. Lifetime results are then compared with those provided by the 1Hz smart battery and the optimum static polling rates presented in Section V.

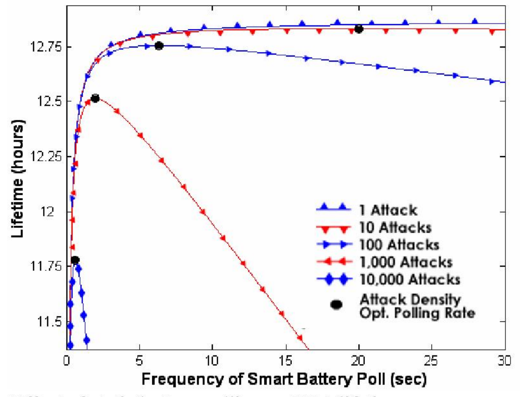

Fig. 2. Effect of static battery polling on PDA lifetime.

A. Theoretical Advantages and Disadvantages

The primary advantage dynamic battery polling rate solutions hold over static solutions concerns network attack density variance. Static solutions have the capability to improve and optimize device lifetime for pre-defined conditions only, while dynamic polling rate algorithms learn from current network activity and apply that knowledge to future smart battery transactions. Dynamic solutions, correctly devised and implemented, should be able to optimize device lifetimes across a vast array of network conditions.

Increased productivity and energy efficiency comes with the drawback of complexity. Future smart batteries will need to alter their hardware to accommodate the new polling rate schemes of battery-based IDSs. For static solutions, this will simply entail hard coding a new value into the system, but for dynamic solutions the change will be more substantial. New smart batteries will need the ability to read a rate value from the SMBus and make the appropriate alterations. The remainder of this section explores the lifetime benefits such a hardware implementation could provide for mobile devices.

B. Algorithm Development and Lifetime Calculations

Motivation for the dynamic polling rate solution was taken from TCP's *slow start* windowing algorithm. The TCP window reduces to a minimum value when contention is detected [18]. Once this has occurred, the window size doubles until it reaches a threshold. Taking a similar approach, the dynamic polling rate solution defines two threshold polling rates. *Min threshold* refers to the minimum time between battery attribute readings, while the *max threshold* refers to the

TABLE I
Network Optimal Polling Rate Lifetimes

Attack Density	**Polling Rate**	**Lifetime in Hours For # Attacks**			
		0	**1**	**10**	**100**
1	0.05Hz	12.678	12.659	12.482	10.711
100	0.50Hz	12.551	12.549	12.532	**12.358**
n/a	1Hz †	12.416	12.415	12.406	12.320
10,000	5.56Hz	11.342	11.342	11.340	11.326
100,000	25Hz	8.6889	8.6888	8.6886	8.6862
		1,000	**10,000**	**20,000**	**30,000**
1	0.05Hz	0.0595	0.0595	0.0595	0.0595
100	0.50Hz	10.625	0.0595	0.0595	0.0595
n/a	1Hz †	11.454	2.7951	0.0595	0.0595
10,000	5.56Hz	11.184	**9.7608**	8.1795	6.5983
100,000	25Hz	8.6620	8.4201	8.1513	7.8826
		40,000	**50,000**	**60,000**	**70,000**
1	0.05Hz	0.0595	0.0595	0.0595	0.0595
100	0.50Hz	0.0595	0.0595	0.0595	0.0595
n/a	1Hz †	0.0595	0.0595	0.0595	0.0595
10,000	5.56Hz	5.0171	3.4358	1.8546	0.2733
100,000	25Hz	7.6138	7.3451	7.0763	6.8076
		80,000	**90,000**	**100,000**	*
1	0.05Hz	0.0595	0.0595	0.0595	*
100	0.50Hz	0.0595	0.0595	0.0595	*
n/a	1Hz †	0.0595	0.0595	0.0595	*
10,000	5.56Hz	0.0595	0.0595	0.0595	*
100,000	25Hz	6.5388	6.2701	**6.0013**	*

† Denotes OEM high-end implemented smart battery polling rate.

maximum time separating battering polling events. Values for these parameters were taken from static polling data. The min threshold was set to the optimal polling rate for devices experiencing network densities of 100,000 (25Hz); the max threshold was likewise set to the polling rate optimal for network attack densities of 1 (0.05Hz). Attack detections cause the current polling rate to return to the min threshold. Battery polling events that do not detect attacks cause the polling rate to be doubled, where the ceiling polling rate is max threshold.

Dynamic polling lifetime simulations were slightly more complex than their static polling counterparts. Attacks detected in static polling rate solutions have lifetime values independent of the precise attack timing, while the attack severity on a dynamic solution is very dependant on the timing of the attack. To model the two timing extremes encountered in networks using dynamic polling rates, equations for *clustered attacks* and *distributed attacks* were developed. Clustered attacks are considered the best case scenario, because attacks are transmitted to the mobile device in a non-interrupted stream. While the device is being attacked, the current polling rate remains at the min threshold. Once the attacks have subsided the device is able to ramp its polling rate up to the max threshold, which is the most power conserving. Distributed attacks, however, are the worst case scenario for devices employing dynamic polling rate solutions. In distributed dynamic attacks, the current polling rate is reduced to the min threshold via the detection of an attack. The next attack is delayed just long enough for the mobile device to ramp up to its max threshold. Timing in this manner allows the attacks to do the most damage, because they disallow the device from remaining at its maximum polling rate for long periods of time. Pseudocode for clustered and distributed attack effects is shown in Fig. 3. Logically, attacks of this nature fall between these extremes.

```
current_mW = total mW provided by fully charged device

while (current_mW > 0)
    if ((attacks_received < network_attack_density AND clustered_attacks)
                                        OR
        (attacks_received < network_attack_density AND distributed_attacks
              AND time_between_polls == max _threshold))
        simulate a network attack
        current_mW = current_mW -
                      (attack_detection_time * active_bus_mW)
        increment attacks_received by 1
        increment attack_time by attack_detection_time
        reset time_between_polls to the min threshold
            (a.k.a. attack_detection_time)
    else
        non_attack_time = time_between_polls +
                      bus_transmission_time
        non_attack_mW = (time_between_polls *
                      idle_bus_mW) + (bus_transmission_time *
                      active_bus_mW)
        total_non_attack_time = total_non_attack_time +
                      non_attack_time
        current_mW = current_mW - non_attack_mW
        double time_between_polls
            (a.k.a. attack_detection_time)
        if time_between_polls exceeds max threshold
            time_between_polls = max_threshold

lifetime = (attack_time + total_non_attack_time) converted into hours
```

Fig. 3. Dynamic optimum polling rate determination algorithm.

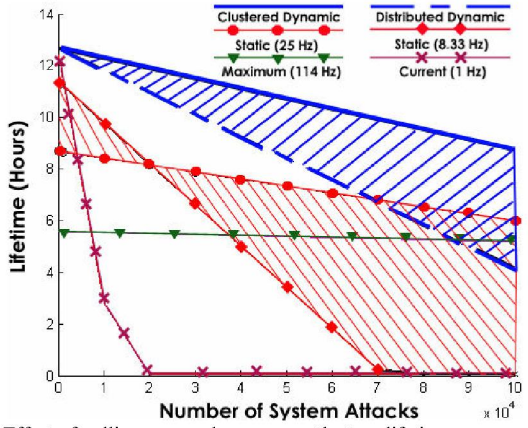

Fig. 4. Effect of polling approaches on smart battery lifetime.

C. Results

An analytical model was constructed in *MATLAB* to calculate the lifetimes associated with mobile devices using the dynamic algorithm. Lifetimes were determined for both clustered and distributed attacks. Results were then plotted, as shown in Fig. 4, for comparison with the most efficient static polling rates, as well as the implemented smart battery polling rate and the maximum polling rate dictated by the SMBus.

The currently implemented smart battery polling rate lifetime performance degrades significantly for even marginally large network attack densities, while the lifetime provided by the maximum polling rate remains extremely stable, though unacceptably low. Static polling rates optimized for network attack densities between 30,000 and 100,000 improve the lifetime of the mobile device significantly, with a marginal drawback of reduced lifetimes for networks with low network attack densities. Finally, the lifetimes associated with the dynamic polling rate perform exceptionally well.

To better analyze the energy conservation achieved by using the dynamic algorithm, lifetime improvement percentages were calculated in Table II. The maximum lifetime percentage increase for static polling represents data collected from a variety of polling rates; this data shows the best case scenario for static solutions, and therefore uses the lifetimes of the optimal polling rate *for each specific network attack density.* The average lifetime percentage increase for dynamic polling averages the lifetimes of dynamically polled devices undergoing clustered and distributed attacks. In almost all

TABLE II
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.

cases, discounting network attack densities of 0 and 1, the dynamic polling rate performed better than the optimal static polling rate. The lifetime percentage increase reaches a theoretical ceiling of 19,000%, when the network attack density reaches 20,000. This is due to the fact that the currently implemented polling rate's lifetime has decreased to an inoperably low value and will no longer decrease proportionally with the network attack density. Because both optimized static and dynamic polling rates maintain reasonable lifetimes after this critical point, their lifetimes will continue to decrease as the attack density increases, thus explaining why the lifetime percentage increase begins to decline once it reaches a network attack density of 20,000.

The theoretical model explored a dynamic battery polling solution and presented a justification for OEMs to enhance smart battery polling capabilities to further support B-SIPS' detection of battery exhaustion, Bluetooth and Wi-Fi attacks. Were these models implemented in future smart battery chipsets, the B-SIPS detection capability could be improved to the point where the attack trace determination method presented in Section VII could be performed on the mobile device without external sampling and post-processing.

VII. Attack Trace Determination Method

It is necessary to acquire high fidelity data that represents a mobile device's instantaneous current usage during a specific type of activity in order to develop an attack trace signature. Due to the limited capabilities of the battery's chipset, an external polling method, utilizing an oscilloscope, is used to provide adequate data. The sampling rate is set to 10KHz with a depth of 15,000 data points per trial, which is significantly faster than network and Bluetooth propagated attacks. A breakout board removes the battery from the device and provides a signal examination platform. Current usage is then determined by monitoring the voltage drop experienced by a precision 1Ω resistor that is placed between the mobile device and its battery, as shown in Fig. 5.

Using *LabView* scripting, the raw time domain data is converted into the frequency domain for examination. A *Hanning window* is applied to the raw data as a smoothing mechanism for the abrupt end of the sample data. Once the Hanning window has been applied, a Fast Fourier Transform is used to convert the signal into the frequency domain. The frequency domain allows device activity to be more thoroughly analyzed by indicating where the majority of the signal resides, as shown in Fig. 6 . However, due to noise from external sources on the amplification circuit, as well as infrequent miscellaneous activity on the device, substantial

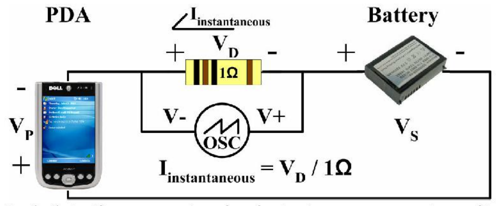

Fig. 5. Attack detection concept, using instantaneous current monitoring.

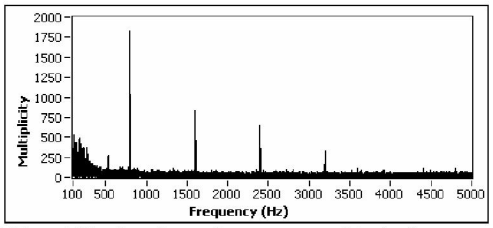

Fig. 6. Unfiltered BlueSmack attack as represented in the frequency domain.

filtering is required to ascertain the important frequencies.

A. Data Filtering and Trace Development

The frequency domain data is filtered in order to remove extraneous data and reduce the large set of (x,y) pairs into a more manageable trace signature. Initial filtering is used to remove all data pairs below 100Hz, as these contain ambient noise picked up by the amplification circuit. This is followed by a removal of all data pairs whose multiplicity falls below a threshold value that was determined by calculating the frequency domain mean. The selection process for ten key data pairs is then handled by using a greatest multiplicity first scheme. The caveat to this is that each spike in the frequency domain is not always a single point, but rather a collection of points with high multiplicity that occur as a result of jitter and interference. To account for extraneous data, a 50Hz window is established around each selected data pair. No new data pairs may be chosen inside a window. Selection continues until either ten key data pairs are selected or there are no remaining selection choices. The resulting key data pairs shown in Fig. 7 are representative of the sampled activity and are cumulatively referred to as a *singular trace*.

To develop a robust representation, 500 trials are compiled into a *complete trace*. This is accomplished by using the first singular trace as the starting point where a population is established around each key data pair. Each of the additional singular traces is added to the initial trace. If a new key data pair falls within the 50Hz area around the original data pair, it is said to belong to that population. If a new key data pair does not fall within any established 50Hz area around the original data pairs, it establishes a new population. After singular traces are combined, the mean frequency and multiplicity is calculated for each population. The ten populations with the largest mean multiplicity are then chosen to represent the activity such that the mean frequency and mean multiplicity of the selected populations make up the new key data pairs. The final step compares singular traces to the complete trace using a modified distance relationship discussed in Section B, creating a self-comparison distribution. A *trace signature* is thus defined as a set of ten key (x,y) data pairs and a

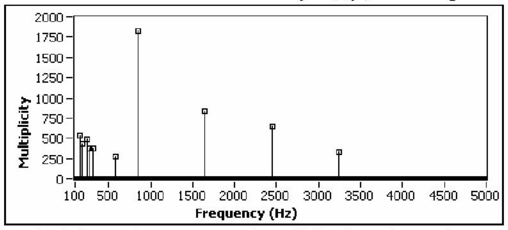

Fig. 7. Key (x,y) data pairs representative of BlueSmack attack trace.

distribution of mean distances between the key data pairs and the datasets used to create them.

B. *Trace Comparison Methodology*

When an unknown activity is detected on a device via a DTC breach, current measurements are taken for the duration of the incident and transmitted to the CIDE for analysis. These raw data measurements are filtered and converted into a singular trace, or trace without any accompanying distance distribution. From this point forward, the singular trace can be compared against the available library of known trace signatures, or traces with accompanying distance distributions. These data pairs represent frequency domain information and are similar to mass spectra, which are commonly used in chemistry to identify the composition of a compound; a similar methodology can be applied to the comparison of traces.

For mass spectra, the most common comparison method is the spectral contrast approach, which measures the distance between corresponding points in the spectra [19]. This approach does not allow for comparing across frequencies, only their multiplicity. To account for activities with different, yet similar, frequencies all possible permutations of comparing data pairs are taken. The permutation with the smallest average distance between frequencies is selected and a modified distance formula is applied to calculate the distance of the data pairs in the frequency and multiplicity dimensions. A mean for the resulting distances is calculated and used in combination with the self comparison distribution of the trace signature. An unknown trace can be said to be similar to a trace signature if the mean distance between their key data pairs belongs to the self comparison distribution of a trace signature within a desired threshold of significance.

Each unknown trace is evaluated against the trace signatures in the library of known activity and attacks by the CIDE. The closest match from the library is evaluated against the statistical threshold, in this case 95%. If the unknown singular trace falls into this confidence interval, then it is considered to be a match. If not, the trace is flagged and stored in an unknown trace signature library containing singular and partial trace signatures. If they are found to match a partial signature, they can be incorporated into the known library once a minimum of 500 singular traces is accrued. At that time the unknown trace signature will be flagged for review and possible admission to the known signature trace library, assuming its representative activity can be identified.

VIII. Bluetooth Attacks

Until recently, many Bluetooth device users considered their systems to be safe from attack. Table III indicates that Bluetooth exploits and emerging attacks are increasing. The ever changing state of attack vectors has opened another avenue for attack signature development, which encompasses the characterization of Bluetooth wireless personal area network (WPAN) attacks. Small mobile computers often have Bluetooth capabilities enabled in discoverable mode out of the shipping box; therefore, devices are vulnerable from the moment they are activated. Moreover, these devices typically lack antivirus software, IDS, and firewall capabilities, so they are unprotected against attack vectors. With new exploits being discovered regularly, a window of opportunity is available for B-SIPS to help protect PDAs and smart phones from attacks. Fig. 8 was adapted from network attack data collected at the CERT/CC by [20] and is overlaid with emerging threats.

TABLE III
Bluetooth Attack Types

Attack Type	Exploit	Threat Level	Focus
BlueSnarf	Authentication	Medium	Information Theft
BlueBug	Hidden Channel	Medium	Root Control
Helomoto	Authentication	Medium	Hijacking
BlueFish	Authentication	Low	Surveillance
Car Whisperer	Weak Passkey	Low	Eavesdropping
BlueSmack	Buffer Overflow	High	DoS
Bluetooth Stack Smasher (BSS)	Buffer Overflow	Medium – High	DoS
BlueSYN *	Buffer Overflow with SYN Flood	Medium – High	Blended DoS
BlueSYN Calling *	Buffer Overflow with SYN Flood	Medium – High	Blended DoS
PingBlender *	Buffer Overflow with Ping Flood	Medium – High	Blended DoS
BlueSniff	NA	Low	Service Discovery
RedFang	NA	Low	Service Discovery
BlueScanner	NA	Low	Fingerprinting
BTScanner	NA	Low	Fingerprinting
BlueJacking	NA	Low	Messaging

* Denotes original attack crafting.

Bluetooth and other blended attacks will become more prevalent as flaws are discovered and exploited. This will offer greater opportunities to develop battery-based trace signatures, since Bluetooth capable devices tend to be of low powered design. Attacking exposed hosts through unsecured WPANs would allow the attacker direct access to the mobile device and its OS environment, completely bypassing any upstream defensive measures. This observation suggests that mobile devices have an increasing need for hybrid IDS protection such as B-SIPS.

While considering plausible attacks that could disrupt PDA and smart phone operations, two original DoS attacks were crafted in the lab: *BlueSYN* and *BlueSYN Calling DoS*. These blended attacks were designed to saturate the test device's multiple communication channels' capabilities to hasten the drain time of its battery resources. Simultaneously attacking the device with a *hping2* SYN flood, affecting the Wi-Fi interface, and a *l2ping BlueSmack* flood, affecting the Bluetooth interface, demonstrates a blended attack that

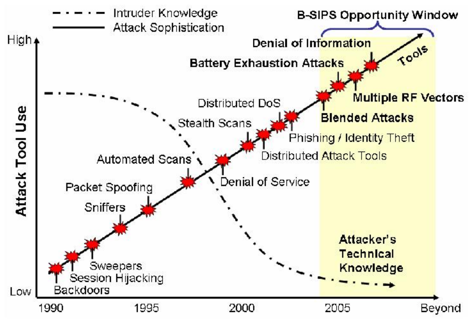

Fig. 8. B-SIPS opportunity window in attack progression environment.

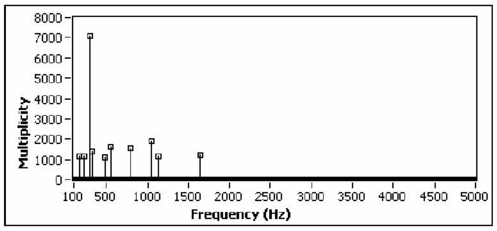

Fig. 9. Crafted BlueSYN DoS attack trace signature for Dell Axim X51.

attempts to saturate multiple communication vectors. The SYN flood propagated through a wired LAN to an access point before finding the target device, while the Bluetooth portion of the attack was launched from a Bluetooth adapter on the notebook computer directly against the targeted device. This previously undocumented attack was named the *BlueSYN DoS*, and its trace is shown in Fig. 9. This crafted DoS attack is considered to be a medium to high level threat because it can exhaust the target device's resources and severely limit both Bluetooth and Wi-Fi capabilities.

For cellular smart phones specifically, extending the above attack and combining it with rapidly dialed phone calls (war dialing) and/or text messages causes the device to react abruptly. This attack requires significant reconnaissance because the attacker needs to know the target device's IP address, MAC address (discovered through fingerprinting), and cellular phone number. This original blended DoS attack was crafted in the lab and was coined the *BlueSYN Calling DoS*. Although unlikely that this DoS attack will be launched outside the laboratory setting, it is considered to be a medium to high level threat because of its resource exhausting and crippling communication effects on the targeted device.

IX. Conclusion

This paper discusses the optimization of smart battery polling, with an aim to increase device lifetime and also increase the number of anomalous attacks that devices can both detect and disable. Optimum battery polling rates were calculated for a variety of network scenarios and their effectiveness was compared with that of the currently implemented 1Hz polling rate. The static solution prevented the device lifetime from being rapidly exhausted but hindered lifetimes during periods of minimal network attack densities.

As an extension of the previously investigated static solution, a dynamic polling rate analytical model was developed and examined. This solution allows devices to maximize their lifetimes for the vast majority of network attack densities, rather than having to select one of two mutually exclusive energy saving schemes, as their static counterparts must do. The dynamic polling rate inherits its minimum polling rate from the optimum static polling rate for a network attack density of 1, or 0.05Hz, and its maximum polling rate from the optimum polling rate for a network attack density of 100,000, or 25Hz. The dynamic polling algorithm informs the device and its battery when to poll next, which is highly dependent on the state of the current network attack density. The dynamic solution will allow future smart batteries to learn about the present state of the device's network and to adapt their polling intervals accordingly. This will, in turn, protect the battery from malicious charge depletion and could also help B-SIPS defend running device applications from being altered, corrupted, or eavesdropped upon.

Finally, B-SIPS research investigated Bluetooth and Wi-Fi attacks to gain an appreciation of possible attacks that could confront PDAs and smart phones. The selected devices were connected to an oscilloscope and battery readings were sampled during various attacks. The sample was then converted from the time domain to the frequency domain, and then further filtered to identify unique (x,y) pairs as indicators of the trace signature. These signatures are stored in the CIDE database and in the future they will be used for attack correlation with *Snort* IDS alerts.

References

[1] T. Buennemeyer, F. Munshi, et al., "Battery-sensing intrusion protection for wireless handheld computers using a dynamic threshold calculation algorithm for attack detection," in *40th Annual Hawaii Int'l Conf on System Sciences (HICSS-40)*, IEEE Computer Soc., Waikoloa, HI, 2007.
[2] Microsoft, "Advanced power management v1.2," http://www.microsoft.com/whdc/archive/amp_12.mspx, 2001.
[3] "Advanced configuration and power interface," http://acpi.info, 2005.
[4] "Smart battery system implementers forum," http://sbs-forum.org, 2005.
[5] "System management bus," http://www.smbus.org, 2005.
[6] E. Thompson, "Smart batteries to the rescue," http://www.mcc-us.com/SBSRescue.pdf, 2000.
[7] F. Stajano and R. Anderson, "The resurrecting duckling: security issues for ad-hoc wireless networks," in *7th Int'l Workshop on Security Protocols*, Cambridge, UK, 1999.
[8] T. Martin, M. Hsiao, et al., "Denial-of-service attacks on battery-powered mobile computers," in *2nd Annual IEEE Conference on Pervasive Computing and Communications*, Orlando, FL, 2004.
[9] R. Racic, D. Ma, et al., "Exploiting MMS vulnerabilities to stealthily exhaust mobile phone's battery," in *15th Annual USENIX Security Symposium*, Vancouver, BC, 2006.
[10] T. Buennemeyer, M. Gora, et al., "Battery exhaustion attack detection with small handheld mobile computers," in *IEEE Int'l Conf on Portable Information Devices (Portable '07)*, Orlando, FL, 2007.
[11] D. Nash, T. Martin, et al., "Towards an intrusion detection system for battery exhaustion attacks on mobile computing devices," in *3rd IEEE Int'l Conf on Pervasive Computing and Communications Workshops (PerCom '05)*, Kauai Island, HI, 2005.
[12] G. Jacoby, R. Marchany, et al., "Using battery constraints within mobile hosts to improve network security," *Security & Privacy Magazine, IEEE*, vol. 4, pp. 40-49, 2006.
[13] In-Stat Research, "Converged devices: smartphones vs. laptops and PDAs in business markets," http://www.instat.com, 2006.
[14] J. Best, "Analysis: what is a smart phone?," in *Mobile and Wireless*, http://networks.silicon.com, 2006.
[15] T. Buennemeyer, G. Jacoby, et al., "Battery-sensing intrusion protection system," in *the 7th Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop*, West Point, NY, 2006.
[16] MSDN, "Microsoft .NET framework developer center," http://msdn.microsoft.com/netframework, 2006.
[17] T. Buennemeyer, T. Nelson, et al., "Polling the smart battery for efficiency: lifetime optimization in battery-sensing intrusion protection systems," in *IEEE Southeast Conf*, Richmond, VA, 2007.
[18] J. Kurose and K. Ross, *Computer networking: a top-down approach featuring the Internet*, 2nd ed.: Addison Wesley, 2002.
[19] K. Wan, I. Vidavsky, et al., "Comparing similar spectra: from similarity index to spectral contrast angle," *Journal of the American Society for Mass Spectrometry*, vol. 13, pp. 85-88, 2002.
[20] J. McHugh, A. Christie, et al., "Defending yourself: the role of intrusion detection systems," *IEEE Software*, vol. 17, pp. 42-51, 2000.