# Battery-Sensing Intrusion Protection for Wireless Handheld Computers using a Dynamic Threshold Calculation Algorithm for Attack Detection

Timothy K. Buennemeyer, Faiz Munshi, Randy C. Marchany, and Joseph G. Tront
*Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061*
*{timb, faiz, marchany, jgtront}@vt.edu*

## Abstract

*This paper proposes a pioneering Battery-Sensing Intrusion Protection System (B-SIPS) for mobile computers, which alerts on power changes detected on small wireless devices, using an innovative Dynamic Threshold Calculation algorithm. B-SIPS enabled hosts are employed as sensors in a wireless network and form the basis of the intrusion detection system (IDS). This detection capability is scalable and complementary with existing commercial and open system network IDSs. B-SIPS implementation correlates device power consumption with IEEE 802.11 Wi-Fi and 802.15.1 Bluetooth communication activity. Irregular and attack activity is detected and reported to the intrusion detection engine for correlation with existing signatures in a database and for forensic investigation by a security manager.*

## 1. Introduction

The primary challenges in developing defensive applications such as intrusion detection systems (IDSs) for small, mobile wireless computers are limited processing capability, memory, and battery power 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. Examining device battery power constraints is an emerging area for security tools for mobile devices in a resource constrained environment. This research initiative is developing a battery-based detection system that employs small mobile devices as sensors that use a power-based threshold algorithm to indicate anomalous activity and trigger alerts.

A good indicator that a rogue process is being run on a device without the knowledge of the user is an unexplained increase in the 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 can provide security administrators (SAs) a complementary tool within a network as a nontraditional method to detect anomalous activity that standard IDSs are incapable of detecting. For instance, if a zero-day or novel attack is being performed, a device enabled with a battery constraint-based IDS could detect an unexpected rise in a mobile device's instantaneous current (IC) usage, whereas the attack may go beneath the thresholds of conventional IDSs and ADSs.

The rest of this paper is structured as follows. Section 2 presents background and related work. Section 3 discusses the system design, the algorithmic approach to B-SIPS, employment of small mobile devices as intrusion sensors, and net-centric server views. Section 4 presents the testing and initial analysis. Lastly, Section 5 provides a succinct conclusion and direction for future work.

## 2. Related work

The security of power-constrained mobile hosts is generally considered as an afterthought compared to service availability. Battery power is an important resource in the wireless domain, especially for small, mobile devices that presents designers with a perplexing problem of choosing more security at the expense of more power usage and potentially less service availability. This is an unresolved tradeoff that continues to challenge network and system developers. Wireless networks are vulnerable to an interloper or eavesdropper who knows how to intercept the radio waves at the proper frequencies [1]. Developing secure

Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07)
0-7695-2755-8/07 $20.00 © 2007 IEEE
1530-1605/07 $20.00 © 2007 IEEE
IEEE COMPUTER SOCIETY
Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on April 20,2023 at 18:23:45 UTC from IEEE Xplore. Restrictions apply.

communication channels through proper authentication could increase service accessibility from a user's perspective but may further increase the device's computational and transmission requirements ultimately leading to faster battery power drain.

An Advanced Power Management (APM) technical specification was developed to better manage device power usage toward extending battery life [2]. APM is an Application Program Interface (API) which allowed computer and Basic Input Output System (BIOS) manufacturers to include power management into their BIOS and operating systems (OSs) that reduced power consumption. The next evolution in power management was the Advanced Configuration and Power Interface (ACPI) that established an industry-standard for interfaces to OS directed configuration and power management on laptops, desktops, and servers [3]. The ACPI specification enabled new power management technology to evolve independently in OSs and hardware while ensuring that they continue to work together. The Smart Battery System Implementers Forum offered an open systems communication standard for industry-wide adoption that described data sharing directly between batteries and the devices they powered [4]. Their introduction of a Smart Battery Data (SBData) specification was used to monitor rechargeable battery packs and to report information to the System Management Bus (SMBus), which implemented a two-wire bus design that could communicate battery data directly to the device [5].

Battery constraint-based intrusion detection and this B-SIPS research endeavor would not be feasible without these technological advances in ACPI and smart batteries. As noted above, interoperability and low power design were inspired by the demand to significantly increase battery life and thus the usefulness of small mobile hosts. Minimizing power consumption is paramount. With the introduction of smart battery technology, the battery pack's embedded electronics can hold SBData, measure battery operating parameters, calculate and predict battery performance, control battery charging algorithms, and communicate with other SMBus devices [6]. Interestingly, the smart battery concept was motivated by the idea that the rechargeable battery would manage its own charging.

Other researchers have investigated extensions to these standard conventions. Benini et al. introduced Dynamic Power Management (DPM) to account for battery constraints [7] [8]. Its goal was to optimize battery subsystem scheduling and management to better satisfy device power requirements, but it failed to address overall consumption.

Certain aspects of IDSs permeate into the wireless domain. Cannady suggested that IDSs be employed together to form a layered defensive approach and that those systems need to use various algorithms to detect security violations, which include algorithms and methods for statistical-anomaly detection, rules-based detection, and hybrids of the two [9].

Nash et al. developed a battery constraints-based IDS for laptop computers toward defending the system against various classes of battery exhaustion attacks of their own design [10]. They 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. They adapted the concept of estimating system-wide power consumption on a per process basis as a method for indicating possible intrusion and identifying rogue processes.

Jacoby developed a Battery-Based Intrusion Detection (B-BID) approach as a purely host-centric IDS solution for mobile handheld devices [1] [11]. This system was comprised of three distinctive IDS applications based on the power capabilities of the device regarding resources and processor clock speeds. At the low power end (fewer resources and slower clock speeds) was the Host Intrusion Detection Engine, which was a rules-based program tuned to determine battery behavior abnormalities based on initialized static threshold levels. In the mid range, a complementary module called the Source Port Intrusion Engine was employed to capture network packet information, during a suspected attack. At the high end, the Host Analysis Signature Trace Engine was used to capture and correlate signature patterns using periodogram analysis in the frequency domain to determine the dominant frequency and magnitude $(x,y)$ pairs. To our knowledge, this system presented the first feasible IDS solution for a small mobile host using battery constraints. However, its deficiencies were twofold. First, it required the user to reinitialize the program to reset the system's state-based thresholds, which could create instances of improper tuning that might allow attacks to slip under the new thresholds or cause unnecessary false positive alerting, depending on the device's activity level. Second, it allowed the device user the option to monitor the system automatically or to manually invoke actions to impede an attack. Although the manual approach is possible, it is unlikely that the device user would monitor the host continuously and be able to respond fast enough to prevent substantial power depletion on a regular basis. As a concept, the B-BID approach presents fertile

Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07)
0-7695-2755-8/07 $20.00 © 2007 IEEE



Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on April 20,2023 at 18:23:45 UTC from IEEE Xplore. Restrictions apply.

ground for further development, scalability, and research extension.

If a small mobile device is kept in a high activity state for extended periods of time, then the battery power will be depleted much faster than normal, decreasing its expected life. Stajano and Anderson suggested the idea of energy depletion attacks as early as 1999, which they described as sleep deprivation torture [12]. 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 [13] [14]. Since system designers of power constrained devices incorporate power management to monitor active processes and to shutdown unnecessary components, sleep deprivation and power exhaustion attacks seek to invade and exploit the power management system to inhibit the device's ability to shift into reduced power states.

In analyzing battery attacks against laptop computers, Martin et al. further subdivided sleep deprivation attacks into three basic categories: service requesting, benign, and malignant power attacks [13]. A service requesting power attack attempts to repeatedly connect to the mobile device with genuine service requests with the intent of draining power from the device's battery. A benign power attack attempts to start a power demanding process or component operation on the host to rapidly drain its battery. A malignant power attack actually succeeds at infiltrating the host and changes programs to devour much more power than is typically required.

An attack of this nature will use more power, and thus demonstrates the need for an integrated battery-sensing IDS. B-SIPS research is developing 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 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 complementary with existing commercial and open system network IDSs. B-SIPS implementation correlates device power consumption with Wi-Fi, Bluetooth, and in the future for cellular phone communication activity. Irregular and attack activity is detected and reported to the online IDE server for comparisons against attack trace signatures. In the future, detection tools must also perform correlation between hosts and robust network-based IDSs.

## 3. System design

B-SIPS research offers a viable model and working system for monitoring power demands that directly affect mobile hosts and can be used to detect attacks and other irregular communications activity. Unusual power usage coupled with network traffic activity can be correlated amongst mobile client devices. This monitoring can lead to early warning and subsequent detection of new or previously *unsignatured* attacks.

The overall design of this system in Figure 1 employs a standard client-server relationship. The server is called the *Correlation Intrusion Detection Engine* (CIDE) server, while the client that reports to it is called the Intrusion Detection Engine (IDE) client. The server performs a multifaceted role in the B-SIPS implementation by receiving near real-time data from the client, interfacing the values into the database, and providing a forensic analysis view of the stored values.


**Figure 1. System design**

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 the Microsoft Mobile 5.0 OS. The B-SIPS suite of tools is initially produced for Dell Axim X51v PDAs running Mobile 5.0 as well as the Dell Axim X50v and X30 running Pocket PC 2003. A beneficial tool during development of the B-SIPS client was the Microsoft Device Emulator, which allowed for rapid prototyping. This permitted programmers the ability to program for various environments, incorporate ACPI members and function calls, quickly debug issues, and to test the functionality of the design. The CIDE server code was tested in a Windows Server 2003 environment.

### 3.1. Impacts of detection systems

Statistical-anomaly detection methods attempt to detect attacks by characterization and analysis of audit logs and a system's behavior to establish a baseline profile of perceived normal system activity. Any activity outside these established system profile norms, such as threshold breaches, is considered to be an intrusion until proven otherwise through forensic

Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07)
0-7695-2755-8/07 $20.00 © 2007 IEEE

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on April 20,2023 at 18:23:45 UTC from IEEE Xplore. Restrictions apply.

analysis. Systems employing statistical-anomaly detection are beneficial because they can detect novel and zero-day attacks without prior knowledge. Any intrusions that are missed are labeled *false negatives*. When normal data activities are misidentified as attacks, they are labeled *false positives* in Figure 2. However ADSs' main drawback is that anomaly detection can generate numerous false positives, which wastes valuable SA investigation time. Ultimately, SA experience with the system determines how effective anomaly detection is at finding malicious events.

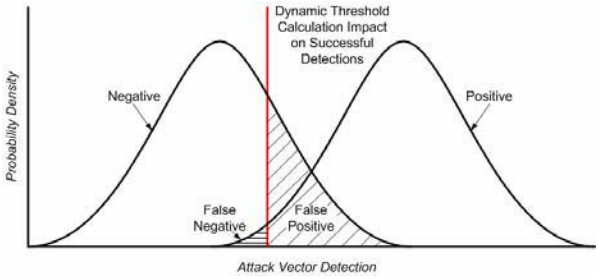



**Figure 2. Theoretical intrusion detection adapted from [17]**

Signature-based detection methods, such as "if-then" constructs, employ signatures to characterize and identify known attacks. Aspects of packet traffic, unique data patterns, and specific audit log entries all provide valuable clues and are often employed in rules-based signature development. Signature profiles are updated regularly as attacks emerge and are identified, so SAs have to habitually maintain their system's signature base. Rules-based IDSs are a mainstay in today's network defenses, as they provide a valuable capability because they consistently detect most known attacks with relatively few false positives.

### 3.2. A hybrid approach

B-SIPS is a hybrid of ADSs and traditional IDSs because it triggers on energy draining events that were not expected using statistical bounds to assess an attack. It also attempts to match power traces of some known attacks and then correlate the attack with other network IDSs. The goal of B-SIPS is to rapidly detect power consumption changes in mobile hosts, which could indicate a possible attack and alert the user and SA of potential malicious activity such as DoS, flooding attacks, viruses, and worms. B-SIPS addresses the hybrid IDS requirements by observing host device battery activity in Busy and Idle states. In the Suspend state, there is no way to measure activity. However, it is possible to measure the number of times a device enters the Suspend state. An attacker, potentially trying to fool the system to subvert the threshold, could use this state changing situation as an attack vector [11].

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 (if available) are transmitted, and how intrusion correlation is conducted. B-SIPS alert notification is done at the client for the user and across the network at the CIDE server for the SA in the system. Certain power-depleting attacks such SYN floods, ping floods, smurf attacks, some buffer overflows, and various DoS 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. Ultimately, a hybrid IDS such as B-SIPS needs to capitalize on the advantages of rules-based approaches to minimize false positives and detect known attacks. Where possible, B-SIPS should detect novel attacks like an ADS without generating false positives that waste SA time.

B-SIPS uses battery constraints and current thresholds to trigger device alerts in Idle and Busy states. The generation of false positives and false negatives is of great concern. B-SIPS attempts to minimize both false positives and false negatives through dynamic threshold tuning, by capturing events, and pairing that data with traffic header information. An attacker with B-SIPS knowledge could potentially trigger a long series of false negatives to intentionally cause power exhaustion on the device.

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. Although false positives are always 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. A tangible goal of B-SIPS is to save valuable SA time by reducing false positive alert investigations.

### 3.3. DTC algorithm description

This research investigated and developed the DTC algorithm as a methodology to detect intrusions based on the battery drainage characteristics of Handheld



IEEE COMPUTER SOCIETY

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on April 20,2023 at 18:23:45 UTC from IEEE Xplore. Restrictions apply.

Mobile Devices (HHMDs). Figure 3 illustrates the DTC sequence of events explained below. Many modern electronic devices and appliances, including HHMDs, are equipped with a Smart Battery System (SBS). The SBS is an enhanced battery pack with added internal electronics that can measure, compute, and store SBData [6]. SBData includes information about the battery such as the manufacturer, temperature, voltage, current, and average current. This DTC implementation used the coredll.dll in the .NET CE Framework to access the system power status structure, containing the HHMD's SBData [2] [16].

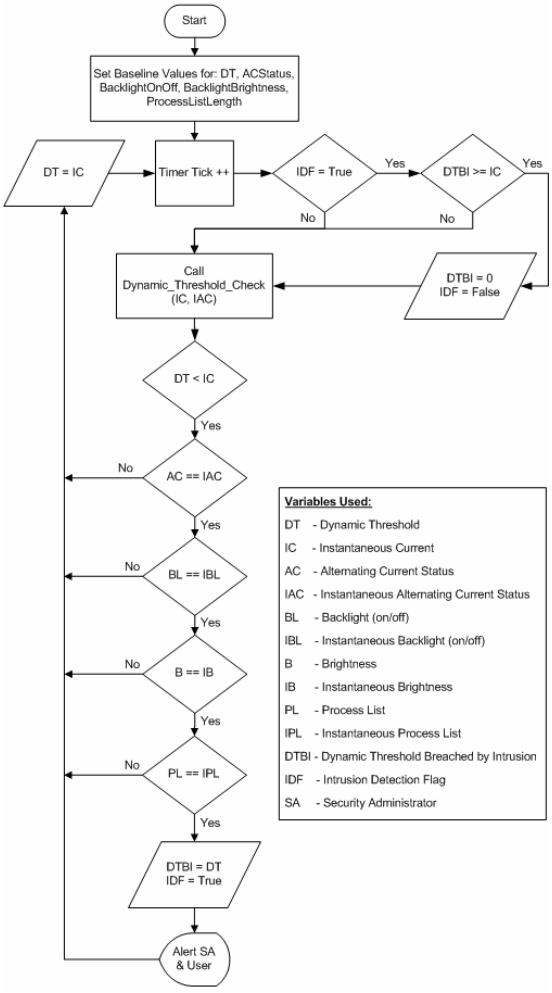



**Figure 3. Logic flow of DTC**

The DTC algorithm employs the IC of the HHMD to determine the Dynamic Threshold (DT). The HHMD battery current usage can increase due to expected activities, including battery charging status, change in the backlight setting, and starting of a new process. Thus, in order to successfully eliminate these potential false positives from the intrusion detection analysis, this system's algorithm has to ensure the difference in IC and DT is not due to known sources.

When the B-SIPS client program initializes, the DT is to equal the IC, which creates a baseline of the different values that affect IC for future comparison. Every subsequent iteration of the client program calls the Dynamic_Threshold_Check() function. This function first checks to see if the IC reading is greater than the DT. If this case is true, it indicates that either user activity on the HHMD such as a process is starting or an anomalous activity that could be an intrusion attempt. The system establishes a baseline and checks whether the HHMD charging status has changed. The ACStatus value derived from the smart battery provides this information. An ACStatus value equal to '1' is analogous to the HHMD being charged, whereas a value of '0' means otherwise. If the HHMD was in a charging state in the cradle, and transitions to a discharging state out of the cradle, then current value will increase. Thus, an increase in current is likely due to the change in battery charging status. If the HHMD was in a discharging state and transitions to a charging state, then there will be drop in the current values to 0mA. Hence, when a HHMD is charging, the B-SIPS client and CIDE server cannot use this algorithm to detect anomalous activity. In either case, an update to the baseline, including the DT value, allows the system to safely reconcile that no anomalous activity occurred.

In the case where the HHMD charging status did not change and the device is in a discharging status, other parameters need to be further investigated that could affect the IC value. To do so, the DTC checks to see if the backlight setting of the HHMD has changed in comparison to the baseline. If the comparison indicates that the backlight was turned on from a previous off setting, this can cause a threshold spike in IC. In this case the baseline, including the DT value, is updated. If the backlight was turned off from an on setting, it causes a decrease in the IC value and does not affect the intrusion detection analysis. If the backlight setting has not changed and is equal to the baseline, further investigation and checks of other parameters that can affect the IC are again needed. If the backlight has been on as compared to the saved device context, then a check to ensure the user did not increase the brightness of the backlight is required. Increasing the brightness can cause a short-term spike in the current and hence could be interpreted to be a false positive. In order to check this parameter, the registry of the HHMD is polled to determine the brightness value. If this value has increased, the change in the brightness value will cause a spike. In this case, by updating the baseline and the DT value, it is assumed that there was no anomalous activity. Decreasing the brightness setting does not cause a spike in the IC. Hence if the brightness has been



IEEE COMPUTER SOCIETY

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on April 20,2023 at 18:23:45 UTC from IEEE Xplore. Restrictions apply.

reduced compared to the baseline or it has not been changed, it is assumed that the spike in IC is due to some other parameters or cause. As the program continues, it updates the baseline and checks other parameters for a reason for the IC increase.

If the DTC algorithm reaches this stage, it has already checked for two of the major factors that affect IC: HHMD battery charging status and the backlight setting. The only other reason for the IC to increase further is if the user starts a new process. The DTC uses P/Invoke to get a snapshot of the running processes. If the total number of processes running is more than the baseline, the system updates the baseline and DT value to account for the change in IC that is due to a new process started by the user. If the total number of running processes matches that of the baseline, that surge in IC is due to an increase in network traffic. The Iphlpapidotnet.dll is employed to obtain the current TCP connections, check for remote addresses with port 80, and reduce false positive alerts related to a user starting the web browser. If there is an active connection to a remote host with port 80, a spike in IC is likely to be a false positive, so the program updates the baseline and DT value. If there is no active connection to a source address with port 80, then there is likely some type of anomalous activity on the HHMD that could relate to an intrusion. At this point, an alert is issued to the SA and the device user of suspicious activity, at which point the SA can utilize the CIDE server's analysis view to investigate the incident. The CIDE server now makes a copy of the reported DT value that was breached and raises the Intrusion Detected Flag (IDF) in the system.

Once the IDF has been raised, on every subsequent iteration the B-SIPS client tries to check whether the IC value has decreased to be less than or equal to the DT value that was breached. This step helps identify the end of an attack event. Once the attack has ended, the system can lower the IDF and repeat the above process to identify subsequent attacks. If the attack has not ended, the system continues to capture network traffic information, such as packet type and source address, and then sends that data to the CIDE server for SA assessment. This information can be used to conduct forensic analysis and to indicate proper counter-measures to the suspicious activity.

### 3.4. B-SIPS IDE clients as mobile sensors

The B-SIPS IDE client reads the SBData and device values of the HHMDs and sends those readings to the CIDE server. Mobile hosts are critical to the overall detection capability, since they generate the data needed by the system to determine if an intrusion has taken place. The IDE client accomplishes this task by calling system functions that read values directly from the system's hardware. The range of values necessary include: time, instantaneous electrical voltage of the battery, IC, battery status, average current milliamperes per hour, physical temperature, and AC line status. All of these values support the DTC algorithm in establishing a tripwire for anomalous activity detection.

In Figure 4a, the initial goal was to read key HHMD values and to display that data for the user. In trying to understand what affects the DTC algorithm, the original design lacked certain factors that were critical in detecting intrusion occurrences. Several factors were added in the IDE client to create a more robust application: the list of processes being run by the host, status of system hardware backlight, and brightness level of system hardware backlight. The present design including these factors is shown in Figure 4b.

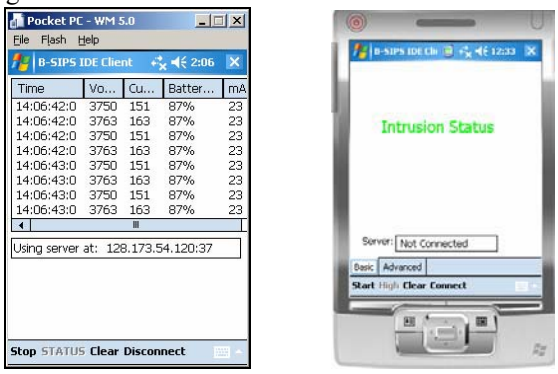



**Figure 4. a) Initial and b) revised basic client**

Beyond calling these values, several other capabilities comprise the IDE client. The first capability is the network communication connection between the IDE client and the CIDE server. The client communicates directly with the server through UDP packets by sending the necessary values retrieved by the IDE client. This allows the B-SIPS IDE client to communicate across different sub-networks, which is applicable in this implementation, using the Virginia Tech campus network. For example, this system could be implemented with the CIDE server residing on the academic network while the IDE clients are able to roam around and through various other sub-networks within the residential area. The system's low-overhead UDP packets are utilizing port 37 on the CIDE server. 08188966</accession-num><urls></urls></record></Cite></EndNote> [13] ADDIN EN.CITE <EndNotding data at a rate of once every second. While this may cause an undue increase in battery consumption, repeated trials have shown that this decrease in battery life is minimal.



IEEE COMPUTER SOCIETY

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on April 20,2023 at 18:23:45 UTC from IEEE Xplore. Restrictions apply.

The second capability of the IDE client is the invoking of the process list in Figure 5a. This ability is important to the DTC algorithm because it aids in preventing false positives when opening new processes. Through the implementation of the P/Invoke library, the IDE client is able to determine which processes are running and to detect when a new process is started. This information is dynamically updated and then factored into the DTC algorithm.

**Figure 5. a) Advanced client and b) alert view**

The third capability is the IDE client's ability to calibrate itself to each HHMD that it operates on. The purpose of this calibration functionality is to compensate for each device's unique backlight, which have differing levels of drain on the system battery. By reading values of the electrical current when the backlight is on, and then reading the values when the device backlight is off, the IDE client is able to calculate a difference value that is used as a factor in the DTC algorithm. This plays an important role when the HHMD switches between Busy and Idle states, which cause the backlight to either switch on or off.

The last of these major capabilities is the ability of the IDE client to run packet analysis. While this is still in development for UDP and ICMP traffic, the current functionality allows the device to capture and relay TCP packet header information to the CIDE. The purpose of this capability is to assist the SA with potentially identifying the attack source and to help expedite necessary countermeasures. This aids the SA in determining what type of attack is being performed or perhaps in detecting a zero-day attack.

The application does more than just read and send values to the CIDE server. The B-SIPS client notifies a network SA and the device user when an intrusion is detected as seen in Figure 5b. This allows the user of the IDE client to take proper countermeasures to start an antivirus program, prevent data from being written to the device, or temporarily shutdown the network connection. This application can also serve as a diagnostic tool for troubleshooting issues involving the system battery. Although B-SIPS is well developed, there are still opportunities for further system enhancements to provide more robust data, to optimize communications, and to refine the detection process.

### 3.5. Correlation intrusion detection engine

The CIDE server is central to the B-SIPS system by processing all information sent from the IDE clients and writing that information into a database for future forensic analysis. This server is able to handle multiple data streams being broadcast from numerous B-SIPS enabled IDE clients.

The CIDE server's data view is shown in Figure 6. This displays the raw data readings being sent from the IDE clients in the order in which they are received. The available data fields in this display include: IDE client identification (IP address), time of reading, voltage, current, battery life percentage, milliamps an hour consumption, physical battery temperate, battery charging status, and AC line connection status. Initially, one drawback of this view was displaying the immense amount of data coming into the server at such a high rate. This made the information difficult to observe and evaluate. Additionally during flooding and DoS attacks, transmission latency was measured at as high as 6 seconds, yet the system was still able to successfully report in a saturated Wi-Fi environment.

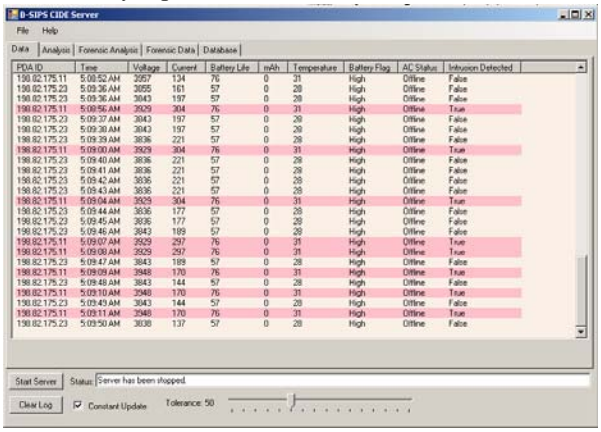


**Figure 6. Data view from CIDE server**

Several solutions were implemented in attempts to make the user interface more intuitive. For example, a control was implemented that permitted the SA to stop the data view from being updated. This allowed the SA to methodically analyze the results of each reading. Another control added to the display was the ability to clear the results in the view. While this ultimately aided in usability, analyzing text values became onerous. This led to the development of CIDE's graphical view.

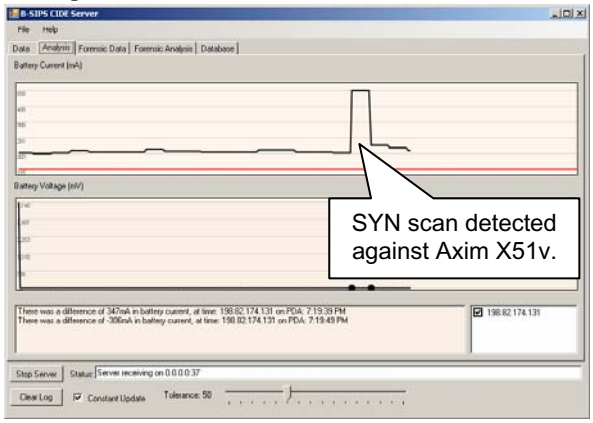
The graphical analysis section of the CIDE server allows for the data received by server to be drawn in a two-dimensional graph. Currently, there are two



IEEE COMPUTER SOCIETY

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on April 20,2023 at 18:23:45 UTC from IEEE Xplore. Restrictions apply.

graphs implemented within this analysis section which is shown in Figure 7. The first graph represents the battery current versus time, and second graph shows the battery voltage versus time. The battery current graph is central to the analysis of the data values, as the IC of the battery is a more reliable indicator of whether anomalous activity is occurring. The other graph plots the instantaneous voltage of the battery. While the voltage of battery is closely linked with the electrical current, its full capabilities are still unexplored, but it is implemented for future system development.

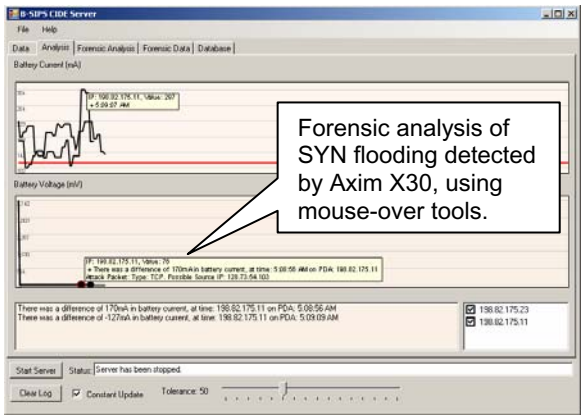



nmap SYN: -sS

**Figure 7. Graphical analysis view of SYN scan**

The purpose of these graphs allows the SA to better visualize in near real-time anomalous activity within the system. For example, a spike in electrical current from one device can usually be a good indicator that an intrusion has occurred on that specific device. The SA can further analyze data from that device to see what type of communications activity has occurred, such as a large load of similar TCP or UDP packets directed at that device. While this information may appear suspicious, if this trend is identified on multiple IDE clients, the SA will have irrefutable evidence that an intrusion is taking place or has already occurred. The graphical view will usually produce congruent spikes in devices affected with similar timing in which the increase of electrical current was detected.

The analysis view of CIDE was implemented, using the C# Graphing Module for Windows Applications [18]. This module well suited the needs of the system and was selected because of its capability to graph numerous data streams. This is vital in displaying the data of multiple IDE client devices and also for future forensics analysis of values in the system's database. An interesting capability of the module that added significant functionality to the system was the ability to graphically insert notifications within the display. This allows the server to quickly alert the SA of problems in an efficient and informative manner. A notification field independent from the graph was implemented to keep long-term track of past security notices over time. The purpose of this field was to prevent past notifications from slipping out of context from the SA. This insures that the SA will have a comprehensive view of the various B-SIPS clients' reported activities.




hping -S -c 10000 -i u100

**Figure 8. Graphical view with forensic tools**

Several capabilities were included to aid in the functionality of forensic analysis in Figure 8, during the development of the graphical display and SA interface of the system. In the deployment of this system with a large number of IDE clients, the graph of the electrical current can become occluded. This can cause analysis of the data by the SA to be difficult. This problem identification led to the development of a field to filter specific IDE clients. Having this field allows the SA to selectively display certain IDE clients to determine if their readings contain suspicious values. The second significant development in this section was the implementation of a tolerance meter. This tuning gauge allows the SA to customize a tolerance range in which those specific values of electrical current are acceptable and do not register as an intrusion or other anomalous activity. The purpose of such an implementation is to compensate for power consumed by the IDE client, other processes, and inconsistencies from the smart battery sensor. Else, the SA could be plagued with numerous false positives.

## 4. Testing and initial results

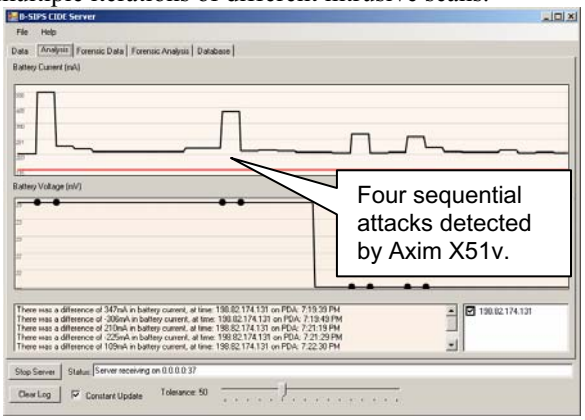
Initial testing demonstrated that the B-SIPS client and CIDE implementation was able to detect several types of attacks. For example, the system was able to detect sequential *nmap* scans. The SYN scan presented a dramatic increase from 209mA to 556mA on the Dell Axim X51v. The timing of the test attacks were closely aligned which would indicate to the SA that suspicious



IEEE COMPUTER SOCIETY

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on April 20,2023 at 18:23:45 UTC from IEEE Xplore. Restrictions apply.

activity has occurred in Figure 9. Each attack produced unique results, yet consistent by type, verified through multiple iterations of different intrusive scans.




nmap SYN: -sS, UDP: -sU, Xmas: -sX, FIN: -sF

**Figure 9. Invasive scans detected by B-SIPS**

In Figure 6, the transmitted PDA reports are assessed by CIDE. The correlation algorithm then compares the attack trace against a signature base and graphically represents the increased PDA power drain to alert the SA, which is displayed as a spike in near real-time on the security manager's console. Using *nmap* and *hping2*, an Axim X51v's IC magnitude attack traces are compared in Figure 10. These attacks launched unexpected packets with no payload at the PDA, but an attack could have been executed using any readily available online attack crafting package such as www.metasploit.com to deliver a payload.

The SA can further investigate the unusual activity by conducting a rapid forensic analysis in Figure 8. The CIDE view provides some basic data mining tools, which show connections between suspicious activities associated to captured packet header data such as:

- Communication Type
- Remote Address
- Local Address (or ESN with cellular devices)
- Source Port
- Destination Port
- Time

Collectively, these datasets can be associated with an anomalous activity to help identify commonalities within and between end unit hosts.

Ultimately, the rapid response by the SA will be the critical enabler of B-SIPS. As with most IDSs, the primary challenge is correlating the activity to determine the event cause. This challenge is not trivial with large scale deployments, so B-SIPS seeks to complement those systems by providing early warning, activity correlation, and a forensic analysis tool set to assist in the investigative process.

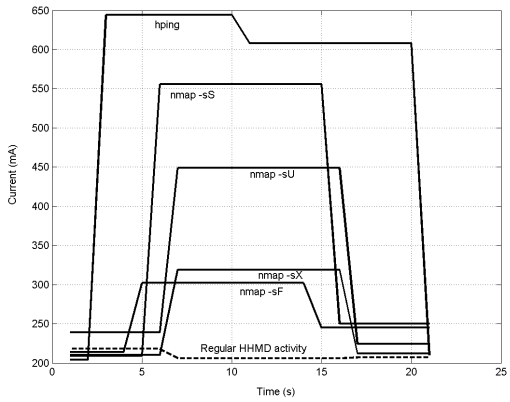



**Figure 10. Magnitude contrast of attack traces**

B-SIPS calculates the DTC value once per second for comparison with the IC, due to limitations in the smart battery chipset. When a threshold breach occurs, B-SIPS transmits its reports to the CIDE server. 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 which could lessen the useful battery life of the device. These tradeoffs for the Axim X30 are characterized in Figure 11, but it is believed that the benefits of pulling reports each second for rapid notification will outweigh the power expenditure.

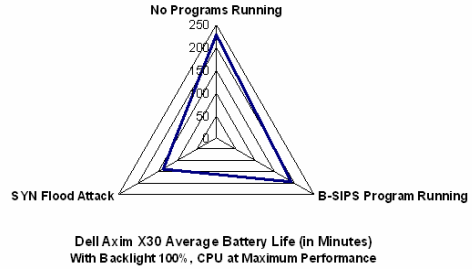



**Figure 11. Average battery life using B-SIPS**

These results show great promise that B-SIPS deployment is feasible as a tool for detecting unusual activity on small mobile hosts during periods of high power consumption. Of concern is the fact that B-SIPS operation on PDAs does come at a price, so some power must be expended to have some level of protection. Based on the testing of Axim X30s, B-SIPS will consume roughly 14% power under normal usage with no attacks and it should save approximately 26% power when the device is under attack. In the future, it is anticipated that DTC and transmission optimizations will provide additional power savings.

Preliminary B-SIPS testing also demonstrated that the battery sensing capability function could capture

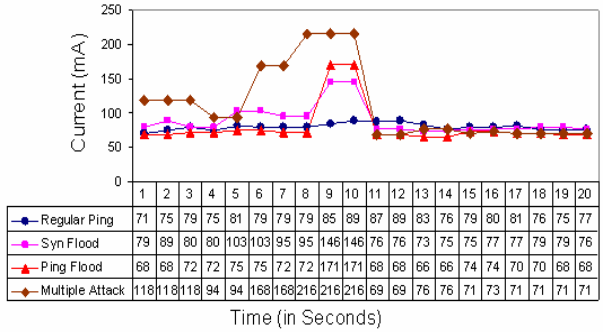


IEEE COMPUTER SOCIETY

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on April 20,2023 at 18:23:45 UTC from IEEE Xplore. Restrictions apply.

the power drain of two PDAs simultaneously. This test used the *hping2* packet crafting tool to generate SYN flood and ping flooding attacks. The attack detection shown in Figure 5b by the B-SIPS client indicates the PDA's DT was violated. The B-SIPS client transmitted the threshold breaches to the server-based CIDE using UDP packets to minimize communication overhead.

Analyzing these results from the different types of attacks against B-SIPS, it can be readily observed that some attacks produced higher power consumption threshold breaches. Another test employed repetitive ICMP traffic to create conditions of high volume Wi-Fi network activity. When the B-SIPS client was attacked using DoS and flooding variants, it was able to detect irregular activity. When a SYN flood was initialized, the battery current would increase considerably for the period of the attack as shown in Figure 12. The electrical current from the HHMDs are plotted versus time to show that future trace signatures could be developed to uniquely identify these attacks.




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Regular Ping | 71 | 75 | 79 | 75 | 81 | 79 | 79 | 79 | 85 | 89 | 87 | 89 | 83 | 76 | 79 | 80 | 81 | 76 | 75 | 77 |
| Syn Flood | 79 | 89 | 80 | 80 | 103 | 103 | 95 | 95 | 146 | 146 | 76 | 76 | 73 | 75 | 75 | 77 | 77 | 79 | 79 | 76 |
| Ping Flood | 68 | 68 | 72 | 72 | 75 | 75 | 72 | 72 | 171 | 171 | 68 | 68 | 66 | 66 | 74 | 74 | 70 | 70 | 68 | 68 |
| Multiple Attack | 118 | 118 | 118 | 94 | 94 | 168 | 168 | 216 | 216 | 216 | 69 | 69 | 76 | 76 | 71 | 73 | 71 | 71 | 71 | 71 |

**Figure 12. B-SIPS response to *hping2* generated flooding attacks and analysis**

The most substantial effects were seen with a ping flood. In this attack, battery current sharply increased, more than it did with the previous SYN flood. It is notable that this surge in battery current lasted for a short period of time and then ended abruptly. Lastly, a combined attack created the largest drain in battery current, along with the greatest duration of time of these experiments. In this attack testing, the battery current shows a distinct range of values. This data can be used to develop power depletion attack traces for analysis and preventing attacks. With good results from B-SIPS for detecting flooding style attacks, the challenge will be to identify viable power-based profile traces for other network and Bluetooth attacks.

## 5. Conclusion and future work

The concept of employing battery constraints as a means of intrusion detection is a relatively new capability that was only recently made possible by developments in smart battery and advanced power management technologies. The B-SIPS design offers a hybrid intrusion detection method that can serve to protect small mobile computers from anomalous activity that seek to drain battery power excessively. This research asserts that small mobile hosts in a net-centric environment can be protected by B-SIPS, which triggers alerts based on power utilization threshold breaches detected by an innovative DTC algorithm. The next step of B-SIPS research will investigate and develop signature traces for Bluetooth and some existing wireless network-based attacks and then continue to improve the forensic analysis tool set.

## 6. References

[1] G. A. Jacoby and N. J. Davis, "Battery-based intrusion detection", IEEE GLOBECOM, 2004.
[2] Microsoft, "Advanced power management v1.2", www.microsoft.com/whdc/archive/amp_12.mspx, 2001.
[3] "Advanced configuration & power interface", http://www.acpi.info, 2005.
[4] "Smart battery system forum", http://www.sbs-forum.org.
[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] L. Benini, et al., "Battery-driven dynamic power management," *IEEE Design & Test of Computers*, vol. 18, 2001, pp. 53-60.
[8] L. Benini, et al., "Extending lifetime of portable systems by battery scheduling", IEEE DATC, 2001.
[9] J. Cannady and J. Harrel, "A comparative analysis of current intrusion detection technologies", In the Proceedings of Technology in Information Security Conference, 1996.
[10] D. C. Nash, et al., "Towards an intrusion detection system for battery exhaustion attacks on mobile computing devices", IEEE PERCOM, 2005.
[11] G. A. Jacoby, et al., "Battery-based intrusion detection a first line of defense", IEEE SMC IAW, 2004.
[12] F. Stajano and R. Anderson, "The resurrecting duckling: security issues for ad-hoc wireless networks", In the Proceedings of the 7th International Workshop on Security Protocols, Cambridge, UK, 1999.
[13] T. Martin, M. Hsiao, H. Dong, and J. Krishnaswami, "Denial-of-service attacks on battery-powered mobile computers", IEEE PERCOM, 2004.
[14] M. Brownfield, et al., "Wireless sensor network denial of sleep attack", IEEE SMC IAW, 2005.
[15] "T. K. Buennemeyer, et al., Battery-sensing intrusion protection system", IEEE SMC IAW, 2006.
[16] "Microsoft .NET framework developer center", http://msdn .microsoft.com/netframework/, 2006.
[17] I. J. Martinez-Moyano, E. H. Rich, and S. H. Conrad, "Exploring the detection process: integrating judgment and outcome decomposition", IEEE ISI, 2006.
[18] T. R. Sidor, "C# graphing module", http://www.mcbyte.dk /default.asp?id=10, 2005.



IEEE COMPUTER SOCIETY

Authorized licensed use limited to: to IEEExplore provided by University Libraries | Virginia Tech. Downloaded on April 20,2023 at 18:23:45 UTC from IEEE Xplore. Restrictions apply.