Facilitating Rapid Response with a Relational Indicator Database Schema and Client Agents

Jason Batchelor

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation
Facilitating Rapid Response with a Relational Indicator Database Schema

and Client Agents

By

Jason Batchelor

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in

Networking and Systems Administration

Rochester Institute of Technology

B. Thomas Golisano College

of

Computing and Information Sciences

October 26, 2010
Rochester Institute of Technology

B. Thomas Golisano College

of

Computing and Information Sciences

Master of Science in

Networking and Systems Administration

~ Thesis Proposal Approval Form ~

Student Name: Jason Batchelor

Thesis Title: Facilitating Rapid Response with an Indicator Database Schema and Client Agents

Thesis Area(s): □ Networking  □ Systems Administration

(circle one)  Security  □ Other ________________________

~ MS Thesis Committee ~

Name                  Signature                  Date

Yin Pan
Chair

Eric Hutchins
Committee Member

Jason Koppe
Committee Member

Luther Troell
Committee Member
I, Jason Batchelor, hereby grant permission to the Wallace Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction must not be for commercial use or profit.

Date: ___________  Signature of Author: ____________________________
Abstract

The threat encompassing the critical computing infrastructure nations depend upon has shifted. A new dynamic of adversaries leveraging a playbook of highly sophisticated, organized, and well funded cyber attacks has emerged. These adversaries penetrate networks using exploits, tools, and techniques that are not detected by traditional client and network security software. Compromised networks stand to lose irreparable amounts of sensitive information and trade secrets if confidentiality is lost. The threat has shifted, but detection and response mechanisms have largely remained the same. They have maintained the same largely ineffective result these advanced adversaries count on.

To counter this, the development and implementation of a client based relational indicator database schema was researched and designed. This schema represents information that, when aggregated over time, signifies an archive of actionable intelligence. The relational model contains tables of client snapshots, each of which are correlated to their respective subset of indicator metadata consisting of differing types of system information. A complete proof of concept implementation was developed using an agent based reporting structure. The agent, named CAITO (Collector of Actionable Intelligence for Threat Observations), reports relevant system information to a database using the developed schema. CAITO is also capable of processing administrative instructions by accessing a remote XML based configuration file. A front-end web portal was also developed to demonstrate the facilitation of analyst queries with the derived dataset. The technical implementation is designed to be integrated into any Microsoft Windows environment. It may be deployed as a Microsoft Self Installer through Active Directory to clients as a Windows based service.
Table of Contents

Introduction .......................................................................................................................... 1
Problem ................................................................................................................................. 1
Review of Literature ........................................................................................................... 2
Document Outline ............................................................................................................... 6
Research and Design .......................................................................................................... 7
Database Design Considerations ....................................................................................... 8
Relational Database Schema Design .................................................................................. 10
  Node Reporting Layout .................................................................................................... 10
  Relational Model .............................................................................................................. 10
  Final Database Design ..................................................................................................... 15
Development Strategy ........................................................................................................ 16
  Schema Development ...................................................................................................... 16
  Database Platform ........................................................................................................... 16
  Analyst Front-end ............................................................................................................ 18
  Architectural Dependencies ............................................................................................ 18
  Agent Development ........................................................................................................ 20
Proof of Concept Considerations ....................................................................................... 22
  Data Reporting Process .................................................................................................. 22
  Reporting Architecture Details ....................................................................................... 23
  Agent Persistence ............................................................................................................ 24
  Security Challenges ........................................................................................................ 25
Results .................................................................................................................................. 29
Agent Software ................................................................................................................ 29
  Program Flow .................................................................................................................. 29
  Class Overview ............................................................................................................... 31
  Dependencies .................................................................................................................. 33
Web Portal .......................................................................................................................... 35
  Functionality .................................................................................................................... 35
Conclusions ........................................................................................................................ 38
Future Work ....................................................................................................................... 39
  Security ............................................................................................................................ 39
  Integration ......................................................................................................................... 39
  Correlative Analysis ......................................................................................................... 39
  Benchmark Statistics ....................................................................................................... 40
Portal Design ..................................................................................................................... 41
Appendices ......................................................................................................................... 42
Acknowledgements

Thank you:

My family and fiancé, without which, I would not have been able rise to this point.

My friends, professors, and academic colleagues for their support throughout the years.

Past and present co-workers and employers, for further developing me professionally and technically.

Special thanks to all my committee members for their guidance and support; Eric Hutchins, Jason Koppe, Luther Troell, and committee chair Yin Pan.
“In the end it all comes down to one thing. You can’t run from the wind, you face the music, trim your sails, and keep going.”

-White Squall, 1996
Introduction

During the first few moments of a computer security breach, responders seek to leverage any available set of resources and tools in order to reduce the exposure of sensitive information and regain control. These tools often process incident related indicators, such as Internet Protocol (IP) addresses, Uniform Resource Locators (URLs), or email metadata, and return related results. The goal for the analyst is to identify exactly which systems have been or are currently being compromised by an adversary. These systems also serve as a potential area to learn more about the tactics, techniques, and procedures of the adversary. The information gathered from affected systems feeds the incident indicator list which in turn, is used to detect and react to adversary presence on the network. The primary purpose of these indicators in the early stages of an incident is to help establish scope. It is through the use of querying different logging capabilities on the network with the appropriate indicators that overall incident scope is established.

Problem

Achieving this result, especially in today’s large scale and globally interconnected networks is extremely difficult. The process is also complicated because many tools provide visibility in one specific area but not others. Some of them may only provide visibility for a certain portion of the network. This forces analysts and companies to invest money and time into other solutions which focus on different areas and use them in parallel to draw correlations for the development of ideas about how serious the incident is.

Many tools responders use focus on the aggregation of network traffic. While this is useful, collection points are typically set up at the perimeter of a network to maximize visibility. This technique is inadequate at aiding in the establishment of scope. A defined incident scope necessitates not only knowing what was compromised, but why it happened, and what the damage assessment was; all of which inform mitigation decisions.

When an undetected adversary compromises an internal system and begins to propagate laterally on the intranet, network visibility is likely limited. The technique fails completely in identifying incidents where there is little or no network fingerprint; such as USB based malware or an insider threat. Today’s perimeter logging mechanisms do not have the capability to process the myriad of particular client indicators, such as MD5s, filenames, or alternate data streams. However, in the described scenarios, host based information is crucial.
In today’s IT environment, the consequences of a security breach have never been higher, and incident responders need a faster, broader, and more effective method to derive incident scope and establish a timeline to understand the incident in its entirety. A logging framework needs to be created for the client, which provides a verbose set of information about the client’s current state.

The purpose of this study will be to identify gaps in client side data aggregation, and fulfill these gaps by developing a relational indicator database schema that is capable of aggregating relevant computer incident metadata. The utility of the schema will be augmented through the use of dedicated, time based, reporting agents.

**Review of Literature**

A survey conducted by Symantec suggests that now more than ever sophisticated and targeted attacks penetrate the security measures on computer networks [1]. These attacks have resulted in the exfiltration of personally identifiable information, such as the 30,000 SSNs that may have been compromised on a college campus [2], or sensitive corporate information [3]. The threat of cyber espionage perpetrated by highly motivated and advanced adversaries cited in the paper entitled “Shadows in the Cloud: Investigating Cyber Espionage 2.0,” published by the Information Warfare Monitor and the Shadowserver Foundation, gives a starting perspective pertaining to the capabilities and focused data collection methods employed by cyber criminals [8]. Documented evidence concerning cyber espionage networks that have proliferated across government, business, and academic networks [8] are of chief concern. The stakes are higher than ever before, and logging capabilities are vital to the identification and remediation of these breaches.

Security Information Management Systems (SIMS) are designed to facilitate searching, data reduction, and correlation of information [7]. They play a major role in identifying evidence of malicious activity. Event identification, aggregation, and correlation methods that are scalable, and depict near real time activity for incident responders continues to be an issue of priority for researchers and corporations. For the analyst, the more verbose a given data set is, the higher the likelihood they will be able to use indicators they find to query for other compromised hosts. It is necessary to dedicate large amounts of storage to the operation of SIMS databases. Much of the data may initially seem too verbose and unneeded; however, the writer’s experience as an incident responder has taught him how inappreciable information captured months ago can quickly become imperative.
A comprehensive SIMS dataset assists in building momentum towards establishing an overall incident scope. When a network intrusion occurs, defenders are placed in a position where they need to catch up to the adversary, to determine what advantage enabled the adversary to progress that far in the intrusion. If the defenders can recreate the entire sequence of events in the intrusion and develop new mitigations around the adversary’s process, which forces the adversary to adjust. On the other hand, if defenders cannot recreate the entire intrusion, any aspect left unmitigated is an aspect the adversary need not adjust. Completeness of analysis drives completeness of mitigations. If the defenders mitigate faster than the adversaries adjust, adversary’s actions become dictated in accordance with the defenders decision cycle. This paradigm shift enables defenders to regain control.

With time and research, SIMS tools will do a better job in fulfilling the needs of incident responders. Mandiant Technologies has recently come out with a device that takes a slightly different approach to the traditional SIMS architecture, but has also driven innovation in the field of host based incident detection and response. The device is called the Mandiant Intelligent Responder (MIR), and it is designed to aggregate alerts that are generated by its agents. The agents are deployed to computers within the enterprise, and are configured to alert on file system indicators which may represent anomalous activity [4].

When designing a reporting architecture, this paper will follow a similar approach -- assigning agents to clients; and then having them report to a collection point. In addition to indicators present on the file system, automated scripts can be assigned to the agents to gather a snapshot of the file system, running processes, and open ports. MIR is one tool that is able to give vital actionable host based intelligence to the incident responder who can then make decisions on remediation and mitigation.

While MIR is a significant step forward, this research proposal aims to develop a time based approach that scales, and is able to consistently aggregate client side information to a centralized relational database. MIR applies a list of known bad indicators and reports which systems contain those indicators. The goal of this project is to design a schema and architecture that builds and stores a comprehensive snapshot of a system across a period of time. This allows analysts to identify approximately when a system was compromised using a specific indicator, and lends the ability to cross reference that snapshot with a snapshot of the system before hand. Doing so may yield insight into other potential indicators that can be leveraged, as well as a complete picture of how an adversary’s compromise matured throughout time.
Scanning using MIR is done on a manual basis [4]. Manually scanning for one period in time will not capture offline clients. This will also fail to capture indicators on systems adversaries have visited, and executed some basic techniques to hide their presence afterwards. A specific indicator might also only exist on a system for a day before deleted. Furthermore, the analysts must know the indicator in advance to find it on the affected systems. The proposed methodology in this paper will increase the chances of the indicator being captured by automating collection and collecting all indicators, thus enabling a better definition of incident scope. The MIR solution is able to gather information on deleted files [13], however, this information can be overwritten on the file system given a reasonable amount of time. The presented solution is more likely to have past records of a clients state since scans and uploads are performed periodically. As analysts uncover new indicators, they can leverage this historical state to identify systems that may not currently have that indicator but did at a previous point in time. Identifying incident scope requires identifying all affected systems, past and present. The matrix below represents some of the fundamental differences in approach with MIR vs. the proposed solution, CAITO (Collector of Actionable Intelligence for Threat Observation).

<table>
<thead>
<tr>
<th></th>
<th>Interval</th>
<th>Visibility</th>
<th>Usage</th>
<th>Historical Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAITO</td>
<td>Automatic</td>
<td>Daily Snapshot</td>
<td>Daily Client Intelligence</td>
<td>Fully Supported</td>
</tr>
<tr>
<td>MIR</td>
<td>Manual</td>
<td>Limited Collection</td>
<td>Reactive Intelligence</td>
<td>Limited</td>
</tr>
</tbody>
</table>

In order to develop a relational database schema that gives a comprehensive view of a client’s state, an examination of what is important to an analyst is essential. There are few realms of computer analysis which share as much symmetry here as the area of computer forensics. By examining what data attributes and elements of the client Operating System (OS) and file system are most vital, one may begin incorporating these ideas into a schema design.

A study conducted by Yinghua Guo, Jill Slay, and Jason Beckett entitled “Validation and Verification of Computer Forensic Software Tools - Searching Function,” assists in identifying client data elements that should be aggregated. The study focuses on providing a framework to ascertain whether a given piece of forensic software can be validated as forensically sound in a given domain. The study provided a hierarchal overview of the important elements of a client node that forensic analysts commonly examine. Some of these areas included registry information, file metadata, and internet browsing data [5]. Further analysis will be needed to determine if these data
points can be aggregated off each client in the fashion and frequency deemed appropriate. However, the research here will help guide thoughts as to what may and may not be worth considering.

Another study entitled “Investigating Sophisticated Security Breaches,” by Eoghan Casey denotes the importance of clear and comprehensive logging throughout the enterprise. The study also asserts that present SIMS solutions such as Cisco’s CS-MARS and nFX from NetForensics, are not designed with enough forensics principles in mind since only some attributes typically stored assist in digital investigations [6]. A schema designed to aggregate client information would aid in facilitating the rapid response of digital investigations. A data point such as virtual memory from clients is noted in the study as being a rich resource of information on compromised hosts [6], and is worth considering adding to the schema.

Currently, adversaries are exploiting the lack of our forensic readiness [6]. The use of a client indicator database to identify an adversary’s tools on a compromised host facilitates a more effective response strategy. A standard IP address firewall block can be simply thwarted by a piece of malware contacting a new command and control address once the original IP times out. At this point, not only is the network still compromised, but the security team has alerted the adversary to their activity, and they may take extra steps to obfuscate their presence on the network. Field research has suggested that sophisticated attackers may migrate hosts to new command and control channels and install new malware, neither of which may be publicly known [8]. The use of a SIMS database designed on client side forensic principles would aid in a quicker identification of compromised hosts, which will allow for a more coordinated and effective incident response.

The creation of a time based agent reporting system that leverages a schema with forensic principles will also assist analysts in the profiling of a system. The paper entitled “Computer Security Incident Handling Guide,” published by researchers at the National Institute of Standards and Technology, states that networks and systems should be profiled at a healthy state to facilitate the identification of malicious changes [7]. Taking this level of profiling down to the client affords incident responders the ability to more accurately determine the time of infection. In addition, it allows analysts to understand what is and is not expected behavior on a client [7]. Automation of client profiling also reduces the need for manual interaction with a compromised host, an action that is discouraged by the study.
Since more advanced and persistent attack styles are being observed as the computing industry matures, reliance on SIMS to identify breaches and the scope of existing compromises will become more and more vital to the overall health of an organization. The latest research and products represent an exciting and innovative trend for the future of SIMS. As attacker profiles shift to the use of more sophisticated and targeted malware, the more thorough and effective defenders need to be with the information they store in SIMS databases.

The collective attitude towards information security needs to be tailored more towards rapid incident response, now. A client based indicator database is the key to establishing the appropriate amount of visibility needed to combat advanced threats. Network compromises are now inevitable; it’s a matter of when, not if.

**Document Outline**

The remainder of this paper is divided into five main areas. Research and design is the first, which explains many of considerations and strategies that needed to be researched pertaining to the database and the proof of concept development. This section also covers the design of the relational database schema and the proof of concept program. The results page is the next main section which gives a detailed overview of the technical results pertaining to the proof of concept. The agent software installed on the client is explained, as well as the web portal template that was developed for analysts use. Conclusions are then drawn and future work is discussed. Finally, the appendix offers a complete view of all programming code and documentation.
**Research and Design**

There are two central elements to this thesis document. The first is the development of a forensically principled relational database schema. The term ‘forensically principled’ pertains to the aggregation of select host based information. Storage of this information is mapped using a relational modeling methodology. The second element of research is the development of a practical proof of concept architecture and its implementation in the enterprise.

Both areas contained their own unique set of considerations. However, decisions regarding one area also had a significant chance of impacting the other. For instance, while considering the aggregation of a particular data set, one also needs to consider how plausible that aggregation might be on an automated and consistent basis, and whether or not that would scale. Also, the data set needs to be aggregated in an efficient, low impact way for the client. Programmatically, it is also desired that methods and functions required to aggregate a certain data set are supported natively to reduce agent size.

Realizing how simultaneously, independent and interconnected these two facets of this thesis is, the writer decided to research and develop them in parallel. This approach lead to a more integrated end result that others will be more likely able to manipulate, and adapt for use in their own environment. It also allowed the writer to make the most of the time he dedicated, by partially mitigating the need to double back and restructure portions of the schema or proof of concept due to an issue pertaining to implementation or feasibility.

This section breaks apart the database schema and the proof of concept portions of the thesis into their respective considerations and designs. Some of the considerations are unique to one specific area, but they will always in some part play a role in the research and design of the other. The first area touched upon is the database schema considerations and design, followed by development strategy that applied to the proof of concept implementation and the resulting design.
Database Design Considerations

At the outset, this thesis focused on two core questions: what kind of information is necessary to gather on the client, and how can this information be structured using a relational database modeling methodology. However, consideration of information gathering naturally led to questions of effective data aggregation and identifying information which is most important. Finally, there is the analyst, the consumer, and the success of this thesis is measured by how well an analyst could use it in the course of an incident response.

Ultimately, the writer needed to first evaluate the fidelity of the information used to aggregate before making any other assessments. The information aggregated needs to be something the analyst can use to pivot on during an incident. An example would be a directory an adversary creates or uses on target systems to pool data for exfiltration or drop tools. Such a finding might yield further insight into what the adversary is targeting or what other file system indicators the incident response team can query on. Rapid response is further facilitated as a result of this.

Value of this information should not just apply towards providing incident responders with actionable intelligence to use directly in incident response activities. It should also help facilitate the feasibility of implementing host based mitigations using other security tools that are deployed. For example, analysis of adversary activity indicates their tactics, techniques, and procedures involve the use of a specific malware specimen which drops executables or dynamic link libraries in a created system directory folder. The incident responder should be able to use aggregate information to make an impact assessment of possibly mitigating files with exe or dll extensions that are written to that given directory. Such activities might be accomplished by developing custom rules for tools such as McAfee’s Host-based Intrusion Prevention System (HIPS).

Ideally, aggregation of all host based indicators would be achieved through the use of one single unified query which captured all possible indicators. However, this is where feasibility and scalability issues come into play. The research aims for automatic scans to be performed on a routine basis, so the ability to consistently store that much information from clients may vary in accordance with the environment the end result is implemented on. Some environments may possess the ability to consistently aggregate this much data to the database server and allow the analysts to query the dataset in a minimal timeframe. Others may implement the solution on a database server with less storage and processing capabilities. Depending on the network infrastructure, bandwidth might also be an
issue if scaled to environments supporting several thousands of computing nodes. Clearly, a flexible and instruction based approach needs to be considered to afford administrators the ability to customize client visibility to a degree due to volume and performance concerns pertaining to the server and network.

Client collection volume is a dual pronged concern, and while it may be partially mitigated by granting the administrator the ability to limit the amount of directory information created, simultaneous reports from thousands of clients presents its own unique array of issues. The primary concern is data collisions occurring on the database at the time multiple clients choose to upload their data. In order to remedy this, some variance will be needed once the appropriate reporting time has been determined by the client. A random back off interval implemented in the client code would help distribute the amount of reports that are uploaded to the database around the same timeframe. Finally, once the issues surrounding the information being aggregated are thought out, considerations made toward the relational modeling of this information must be made.

Relational modeling is beneficial because it allows for transparent correlation of data between two different data sets. The Database Management System (DBMS) governs the relational mapping of data, so long as a schema definition is present. As the writer began assessing how to model the different types of information aggregated from the client, it became clear the end result would emulate a combination of hierarchal and relational modeling. Each client will have one or more snapshots associated with it, which together represent the overall scan. The snapshot will contain related tables which contain more detailed information concerning data tables and attributes. Many of the subsequent relationships between tables will be one-to-many relationships. One snapshot will have many directories enumerated, each directory will contain many files, and each file may contain one or more alternate data streams. Once the necessity of combining hierarchal and relational modeling methodologies into the overall design was realized, research began for the actual design of the database structure.
Relational Database Schema Design

After all relevant issues regarding the creation of the database schema had been identified, crafting the overall table mapping and structure began. The general node reporting layouts were established so that each client could be uniquely identified and possess multiple snapshots (or scans). Individual snapshots contain more granular information captured in relational tables represented in a hierarchal fashion. Once the mapping was complete, a relationship diagram was created using Microsoft Access, and was also used to upload the schema definition to server running Microsoft SQL.

Node Reporting Layout

In deciding how to map clients and subsequent snapshots effectively, the writer incorporated data pertaining to them both into a single parent table. The snapshot table is the parent table which maps to sub tables in the relational hierarchy containing state information of the client. They are taken over a pre-defined, yet consistent, time period. Primary and foreign key constraints are automatically created to uniquely map data tables passed from each client over time.

All clients are uniquely defined using the MAC address that they use to report. That attribute is used as the identifier of unique clients in the snapshot table. So a client with a given MAC address can be queried for in SQL to see what corresponding snapshots it has generated. The client hostname and reporting IP address is also recorded in the snapshot table to better facilitate client identification. Pivoting on MAC address alone is not necessarily a reliable enough indicator, since network cards get replaced from time to time. Hostname and IP addresses may also be used as possible identifiers to zone in on a particular client. This approach was found to be the most consistent and effective way to grant analysts the ability to map unique client identifiers, while taking into account that one or more of these parameters may change over time.

Relational Model

The following section seeks to describe each of the tables with respect to the attributes they possess and their relationship mappings. A figure representing the design layout of each table is presented initially, along with its description.
Snapshots Table

The Snapshots table is the parent table in the entire relational hierarchy. It maps directly to the following tables: ProcessList, Directories, ActiveTcpConnections, TcpListeningConnections, and UdpListentingConnections. All relationships from the Snapshots table to other child tables are one-to-many. The SnapID attribute is the primary key for this table and it used as the foreign key in all sub tables. SnapDate represents the date and time that that client began generating the report for upload. ClientMAC is the unique MAC address assigned to the first active IPv4 enabled Network Interface Card (NIC) in the client’s hardware profile. The IPAddress attribute represents the clients IPv4 address and is derived using the same method as the ClientMAC attribute. The Hostname attribute represents the computer name given to the client by the Windows Operating System (OS).

ProcessList Table

The ProcessList table is the child table of the Snapshots table and a parent of the ThreadList table. It maintains a many to one relationship to the ProcessList table, and a one-to-many relationship to the ThreadList table. It inherits the SnapID attribute as a foreign key from the Snapshots table, and uses ProcID as its primary key.
identifier. The PID is the unique identifier associated to a given process by the client when the agent scan was invoked. ProcessName is the name assigned to the process, and ProcessOwner represents the client user the process is running as. StartTime represents when the process was invoked and CPUPTime shows how much total time was used by the client CPU when processing instructions. PhysicalMemSize represents the associated physical memory usage for a given process. Finally, Threads shows how many threads are running under the given process.

**ThreadList Table**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProcID</td>
<td>Number</td>
</tr>
<tr>
<td>ThreadID</td>
<td>Number</td>
</tr>
<tr>
<td>StartTime</td>
<td>Text</td>
</tr>
<tr>
<td>CPUPTime</td>
<td>Text</td>
</tr>
<tr>
<td>Priority</td>
<td>Number</td>
</tr>
<tr>
<td>StartAddress</td>
<td>Text</td>
</tr>
</tbody>
</table>

Figure 3 – ThreadList Table

The ThreadList table is a child to the ProcessList table, and has no sub tables listed under it. It maintains a many to one mapping to the ProcessList table. The ProcID value is inherited from the ProcID table as a foreign key value. This table uses no primary key. The ThreadID is the identifier the client associates with a given thread. StartTime represents the time when the thread was invoked, and CPUPTime denotes how much time the CPU has dedicated to processing instructions on the threads behalf. The Priority attribute contains the corresponding priority value the client gave the thread. StartAddress displays the starting address the thread used when invoked,

**Directories Table**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirectoryID</td>
<td>AutoNumber</td>
</tr>
<tr>
<td>SnapID</td>
<td>Number</td>
</tr>
<tr>
<td>DirectoryName</td>
<td>Text</td>
</tr>
</tbody>
</table>

Figure 4 – Directories Table

The Directories table is a child to the Snapshots table and a parent to the Files table. A many to one mapping is made to the Snapshots table and a one-to-many mapping is made to the Files table. It inherits the SnapID
value as a foreign key value and generates its own primary key value of DirectoryID. The DirectoryName value is the directory path that was enumerated on a given client.

**Files Table**

The Files table is a sub table of the Directories table, and a parent to the AdsData table. To the Directories table a many to one relationship is drawn, while a one-to-many relationship is drawn to the AdsData table. The Files table inherits the DirectoryID value as a foreign key value while generating its own primary key value of FileID. The FileName attribute denotes the name given to the file from the client. LastWriteTime, LastAccessTime, and FileCreationTime denote the file Modified, Access, and Creation (MAC) times. FileSize represents how large the examined file was in bytes. FileMD5 is the unique MD5 hash value generated for that given file. Finally, FileADS is a Boolean attribute used to determine if a file possesses an Alternate Data Stream (ADS).

**AdsData Table**

The AdsData Table is a child of the Files table and has no child subtables of its own. It has a many to one relationship to the Files table where it also inherits the foreign key value of FileID. The attribute ADSPath denotes the full path and name of the ADS. ADSSize denotes how large the ADS is on the client file system in bytes.
ActiveTcpConnections Table

![ActiveTcpConnections Table](image)

ActiveTcpConnections is a child table of the Snapshots table and has no child tables. The relationship ActiveTcpConnections has to Snapshots is many to one, and it also inherits the SnapID value as a foreign key. There is no primary key value for the ActiveTcpConnections table. LocalTCPConnection denotes the client address and port number used on the client for the connection. RemoteTCPConnection holds the remote IP address and corresponding port number. TcpConnectionState gives insight into the status of the existing connection.

TcpListeningConnections Table

![TcpListeningConnections Table](image)

TcpListeningConnections is a child of the Snapshots table with no sub tables. The relationship the table has with Snapshots is many to one. It inherits SnapID as its foreign key value and maintains no primary key of its own. TCPEntry lists the client IP and port number for the given connection.

UdpListeningConnections Table

![UdpListeningConnections Table](image)

The UdpListeningConnections table is a child of the Snapshots table and does not have any sub tables under it. This table maintains a many to one relationship with the Snapshots table and uses SnapID as its foreign
key. There is no primary key value for the UdpListeningConnections table. The UDPEnd attribute holds the IP and port information for a certain connection on the client.

**Final Database Design**

The chart illustrated in Figure 10 below, depicts all nine tables that are part of the client indicator database as well as their containing attributes. This diagram was generated using MS Access, as was most of the initial mapping of the schema.

![Relationship Chart of Client Indicators](image)

**Figure 10 – Relationship Chart of Client Indicators**
Development Strategy

When making high impact decisions pertaining to programming language, tools, and capabilities to make use of, the writer needed to consider the chief target for the proof of concept. Since most security concerns are directed toward Microsoft operating systems and its line of products are the most widely deployed in commercial enterprises, it made the most sense to target clients running some variant of the Microsoft Windows OS. The decision to develop for these systems had high impact pertaining to the options and capabilities the writer had at his disposal. When deciding on development languages and tools to leverage, the author primarily went with solutions developed by Microsoft due to their tightly integrated product line and compatibility.

Schema Development

At a high level, the creation of the database schema was done using MS Access. The software offered an easy and straightforward way to create data definitions and illustrate how they are interconnected with each other. Changes made to attribute definitions in tables could be done with relative ease using the design view, and tested using sample data sets.

Perhaps the largest advantage found when using this software was the ability to upsize schema changes to an MSSQL database on the fly by using the MSSQL wizard in MS Access. This enabled a quick way to make schema changes and then replicate them to the MSSQL database. When troubleshooting issues pertaining to incorrect attribute or relationship definitions, this functionality is vital.

Database Platform

Through observing the capabilities uncovered in the MS Access software, exploration began pertaining to the viability of MSSQL Server as a database hosting solution. The transparent support for MS Access table upsizing was a large plus from a schema development perspective. When considering the clients need to upload its scan results to the database, the writer needed to consider capabilities MSSQL supports that facilitate this.

Data content is able to be uploaded en masse using a supported Component Object Model (COM) referred to as XML Bulk Load. When seeking to evaluate the utility of this capability and what other additional requirements would be needed, the following excerpt from the Microsoft’s MSDN library was instructive [9].

“The Execute method of the XML Bulk Load object model takes two parameters:
- An annotated XML Schema Definition (XSD) or XML-Data Reduced (XDR) schema. The XML Bulk Load utility interprets this mapping schema and the annotations that are specified in the schema in identifying the SQL Server tables into which the XML data is to be inserted.

- An XML document or document fragment (a document fragment is a document without a single top-level element). A file name or a stream from which XML Bulk Load can read can be specified.

XML Bulk Load interprets the mapping schema and identifies the table(s) into which the XML data is to be inserted.

If utilized at the agent level, an accompanying schema definition file would need to be provided with the resulting data from the scan before the data upload could take place. This means that some sort of schema definition file would need to be developed so the database would understand the data mappings of the system scan. If this methodology was adopted, the client scan file would also need to be generated in XML.

Another option was using the INSERT statement to upload XML data into the SQL database using an OPENXML function. However, Microsoft states that using the Bulk Load utility lends better performance results when inserting large volumes of data [9]. Ideally, one simultaneous upload would be best from an efficiency standpoint. Less uploads from the client leads to a decreased chance of data collision on the database server. It also allows for one unified upload process, which will be easier to control. This control will be critical when implementing random back off sequences in client upload to mitigate data collisions and network congestion.

Upload capabilities for MSSQL are supported in C# and the ability to query this data is supported through the SQL language itself. However, a usable front-end provided to the analyst is a reasonable requirement since it is unreasonable to expect that analysts should query the database directly. MSSQL supports multiple web languages to serve as front-ends for analyst’s queries. So there are few restrictions in the area of web development when going with MSSQL server.

The ability to change schema definitions quickly through the MSSQL upsizing wizard in MS Access, is helpful from a development and testing perspective. The Bulk Upload support capability the C# programming language has for MSSQL has the trade off of providing an additional schema file once the database upload takes place. However, it provides a better performing methodology to use when uploading large amounts of data. It also provides for better control of client uploads. Although MSSQL is proprietary piece of software, there is support for
an array of open web programming languages that can be leveraged for front-end web development. Due to these advantages, MSSQL server made the most sense to utilize as a database platform.

**Analyst Front-end**

For the proof of concept, a demonstration of the potential front-end capabilities a web platform could leverage to display and manipulate aggregate data was desired. There were two main approaches examined when seeking to decide on a language combination to develop a sample front-end: HTML/PHP and ASP.NET/C#.

The use of an HTML front-end with PHP to query the backend database represents the most open potential since the development and hosting of the front-end can be done on free and open source platforms such as Apache Web Server. ASP.NET/C# would require me to use the Visual Studio IDE and host on IIS, both of these solutions are not free, but offer a great degree of usability and native support with MSSQL. With respect to ASP.NET/C#, the ability to retrieve and display database information in a visually pleasing manor was seamless and can be accomplished through the use of one integrated IDE.

One of the most attractive aspects of ASP.NET/C# was the ability to quickly develop SqlDataSource query objects and place them into the web page layout. This facilitates real time data feeds from the database, so it would be possible to easily display to the analyst statistics such as: how many reporting clients they are, how many snapshots are stored in the database, and the total amount of Alternate Data Streams identified. ASP charts can also be mapped to SqlDataSource objects, so graphs illustrating data trends can be easily rendered. Also supported is the use of grid view layouts to display the results of analyst database queries.

The cited reasons offer a high degree of usability and native support when compared to the alternative of HTML and PHP. For these reasons the writer decided to use ASP.NET and C# for front-end development. The usage of both of these languages was available in Microsoft Visual Studio 2008. The IDE offers a simple way to host and troubleshoot the web solution.

**Architectural Dependencies**

The agent based reporting solution follows the client server paradigm, so typical dependencies regarding uptime and connectivity are applicable. Clients with the agent installed will reach out to a predefined server to either
retrieve or store information. The agent will need to connect to the database server to initiate the upload; it will also need to retrieve information such as the schema definition file and an instruction file with reporting details.

Regarding the client upload, a database server running MSSQL as the database platform is required. Primarily due to the way data is uploaded from the client to the server using the MSSQL Bulk Upload COM object. This is the only true dependency that needs to be addressed; however, the client needs to know where to store the information and this information needs to be retrieved from an instruction file.

An instruction file containing details for the client to parse when invoked will need to be hosted. These specific attributes are discussed in greater detail later in the document, but they primarily concern information regarding where to report information, when to report it, and what information to aggregate. The client needs to be able to parse this information in a unified and supported manor. The eXtensible Markup Language (XML), provides a standard framework, where elements containing attributes can be processed. Parsing XML documents is also supported in many object oriented programming languages since it is a well accepted standard for representing data. Due to these reasons, the writer concluded it best to host the configuration file in this format. The client will need to retrieve the configuration file from a web server. Microsoft IIS was chosen as the hosting web server for the configuration file since it was used to develop the analyst web front-end.

The common schema definition file will also need to be hosted for the client to retrieve. Before the agent uploads the results of its scan to the database, a schema definition will need to be provided to the bulk upload method. The XSD file will be hosted as a web download for agents when its primary method is invoked. IIS was used to host this XSD file since it was already being utilized for web development and the hosting of the XML based configuration file.

Reporting is primarily accomplished using the client server paradigm, and a centralized database server is used to aggregate this information. Two files will need to be hosted for client retrieval, the XML based configuration file and the schema definition file. Figure 11 illustrates the architectural dependencies required for the proof of concept.
Agent Development

Development of the agent needed to be in an object oriented programming language which took advantage of a framework with both backwards and forwards compatibility, which is a standard part of any Microsoft Windows operating system. It also needed to support the Bulk XML upload component which uploaded from the client to the central database.

The .NET framework in Microsoft is a standard part of any Microsoft Windows installation, and is a requirement for many of the applications which run on Windows. The C# development language is a highly supported solution by Microsoft which utilizes the .NET framework and supports the Bulk XML upload COM...
object. Programs developed in C# using the Visual Studio IDE can be compiled into Microsoft Self Installer (MSI) files.

MSI files are useful for enterprise deployment via Active Directory. For an agent-based solution intended to be deployed on Windows clients, this is a tremendous advantage, since most enterprises with MS Windows clients use Active Directory for client management and authentication. MSI files are also highly configurable even when compiled. For example, registry entries that are assigned values as a part of an install can be changed in the MSI file itself using Microsoft tools such as Orca. This capability affords administrators the ability to manipulate these values before enterprise deployment. The MSDN library provides the following description of Orca’s capabilities [10].

“Orca.exe is a database table editor for creating and editing Windows Installer packages and merge modules. The tool provides a graphical interface for validation, highlighting the particular entries where validation errors or warnings occur.”

The integrated capabilities C# affords programmers using the Visual Studio IDE are extremely beneficial during the development phase. Administrators of Windows environments will also appreciate the customization options afforded to them through the use of MSI editing tools such as Orca. Considering the many advantages offered by C#, the decision was made to use this programming language to write the agent.
Proof of Concept Considerations

The considerations pertaining to the proof of concept portion of this thesis required careful and calculated decision making, with ample amount of attention going towards items such as data reporting processes and overall reporting architecture. Once the writer decided on a reporting architecture there were several challenges needing consideration pertaining to usability and security. Local agent error handling was also something that needed to be dealt with appropriately on the client. Achieving effective error handling means careful consideration of all the things that could go wrong at and during run time. Anticipating security concerns that might arise when the agent is deployed is also paramount. The agent needs to be robust, and withstand attempts at sub version by the adversary.

Establishing overall program flow and architecture are the first two objectives discussed in this section. Next, an identification of the architectural and security challenges are made and described further. Finally, a section that speaks to considerations given towards establishing overall agent persistence on the client is given.

Data Reporting Process

The intention when this proof of concept began was for the client to upload the results of their system scan to a centralized database using the schema definition previously created. The process by which this is done falls on the agent itself once a system scan is complete. At a high level, the reporting process is broken down into three phases: schema retrieval, generation of the client scan file, and data upload.

When invoked, a GET request is generated from the client to the web server hosting the schema file. The file is downloaded by the agent to the local client file system. Once the download is complete, the schema file will be used to partially fulfill the requirements needed in order to upload data to the relational database. The schema definition file provides the mapping for all data generated from the client to the relational database.

Generation of the system XML file is the next step in the process. The agent makes calls to subsequent classes to aggregate client information, and this information is then stored within the appropriate XML element. Retrieval of system information is done in parallel with the creation of the XML based system file. Once the system XML file is complete, it is written to the file system in the programs working directory.

Finally, with the retrieval of the schema definition file and the generation of the system XML state file, the database upload method is invoked. A separate thread needs to be created in order for this to happen, and once done
the XML SQL Bulk Upload COM object can be used to upload all the system data. This upload is done in one procedure call, and once complete, the system XML file and schema XSD is deleted from the file system.

**Reporting Architecture Details**

The agent will retrieve reporting instructions through accessing a XML based configuration file hosted on a web server. The configuration file will contain XML elements with the following information: the time for the client to report, the address of the database server, the name of the database, and the root directories that the client should aggregate files and file metadata from. This approach allows administrators to adjust client report volume with respect to the file metadata that is aggregated. The reporting time can be adjusted to compensate for times of high bandwidth usage or for maximum client visibility. Database name and address can also be changed on the fly to compensate for infrastructure changes or unexpected server malfunctions. While some configuration options are retrieved through the access of a hosted configuration file, there are some local configuration parameters that need to be supplied.

There are two main parameters that are defined as registry values once the program is installed. The first is the address of the web server hosting the configuration file, and the second is the name of the configuration file itself. These attributes are defined in the local registry of the client once the agent is installed, however, that does not mean that they cannot be customized by administrators prior to deployment. As stated, since the installer is an MSI file, the Orca tool developed by Microsoft as part of the Windows SDK Components for Windows Installer Developers package, can be used as an editor to modify these attributes [10]. Figure 12 illustrates the usage of the Orca tool to accomplish this.
Figure 12 – Orca Tool Modifying Program Registry Values

The final element that impacts the overall architecture is the schema definition file, which is statically named “UniversalSchema.xsd”, and stored remotely on a web server. The file cannot be stored locally on the client for one central reason. The schema definition file, if modified, could corrupt the overall data upload. Hosting this file in a central repository partially mitigates this from occurring.

Agent Persistence

The program is ran as a service in Windows, which is advantageous because the .NET framework has a Windows service structure which integrates seamlessly into the design. The service can be viewable through Windows in “Services and Applications.” By default the service can be automatically started once installation has completed and a system reboot has taken place.
While persistence is established natively though the use of the Windows service framework, errors and issues in the agent itself need to be appropriately handled to ensure the program stays operational. Potential errors in the code were handled with try and catch statements where the appropriate exception was logged if encountered and written as an event to the application portion of the Windows event viewer.

Some errors are dealt with transparently where the overall state of the program is unaffected, while others may force the program to abandon the current runtime function and revert back to the main method. An example of an issue handled transparently would be a directory that was specified to be aggregated from the client that did not exist on the local system. If this exception is encountered, the directory is simply discarded and the program continues unabated. However critical errors, such as the inability to retrieve the schema definition file, are recorded and the program reverts back to the main method where it will wait until its reporting time before reattempting.

Agent persistence is very important to the overall client visibility the solution provides. It is crucial that potential anomalies be accounted for and dealt with gracefully. Non critical errors are documented without much impact to runtime, while high impact errors are documented and a reversion is made to the main method.

**Security Challenges**

Security issues surrounding the discussed implementation revolve around the agents data upload process and some potential architectural vulnerabilities. By attempting to proactively identify areas of concern, the software can be developed from the ground up with security threats and advanced adversaries in mind. The perpetual argument of usability versus security is certainly applicable with the challenges identified. Multiple solutions are presented to potential problems identified.

One of the concerns considered early on by the writer was the potential for an adversary who has successfully compromised a client PC, to try to grab the registry values containing location of the remote web server an agent queries for instructions. An adversary with sufficient privileges on a compromised client may be able to view or modify this setting. Through the viewing of this registry value, and adversary could potentially try to connect to the remote web server to view the file. By viewing the file they could gain a greater understanding of directories that are monitored as well as the reporting infrastructure.
To partially mitigate the potential of this occurring, the web server hosting the configuration file should be doing so via secure HTTP. By doing so, conventional HTML get utilities the adversary may pull down to the machine will need to be compiled with SSL support enabled. In the interest of time and scope, the development of the proof of concept focuses on the retrieval of the configuration file over standard HTTP, however enabling this functionality is relatively trivial.

Denying the use of some command line tools is another way to inhibit adversaries. One way this could be accomplished is through the use of proper security precautions implemented in Active Directory. By implementing the group policy object entitled “Prevent access to registry tools,” and setting the value to enabled, users are disallowed from using the command line tool regedit. Figures 13 illustrates the effectiveness of implementing this policy by showing the results pertaining to the command line utility regedit.

Denial of read and write access to registry elements for standard domain users is the best and most comprehensive mitigation against adversaries who seek to subvert the functionality of the proof of concept. The measures taken in the group policy application described above do not mitigate on command line utilities such as “reg”, which can be used to query specific registry values. Therefore, setting the proof of concept registry entries to not inherit permissions from the objects parent, and removing standard user groups from the permissions list will
inhibit domain user accounts that may be under adversary control, from querying these registry values. Figure 14 shows an example of this configuration.

![Advanced Security Settings for SIMSClient](image)

**Figure 15 – Appropriate Permissions for Registry Folder**

Such changes can easily be made on a mass scale through Active Directory. These permission changes do not impact the program runtime in any way since the process runs under the SYSTEM account in Windows. Figure 16 illustrates the result of these permission changes, when an attempt to query the registry key is made.

```
C:\Users\Jason>reg query HKLM\Software\SIMSClient
ERROR: Access is denied.
```

**Figure 16 – Result for User Attempting to Enumerate Key Information**

By implementing these permission settings, adversaries are inhibited from leveraging standard command line tools with the accounts they compromise. They are also inhibited from leveraging Windows API calls in scripts which do similar tasks. This is because the adversary is still running their script as the unprivileged user, who will
not have appropriate permissions to access the key values. So long as appropriate permissions are associated with
user accounts in the enterprise, potential damage to the functionality of the proof of concept and unauthorized
intelligence gains by the adversary should be largely reduced.

For the proof of concept to remain an effective intelligence gathering solution, it is imperative that
intelligence pertaining to aggregated directories remains undisclosed. Utilizing appropriate permissions to client
keys that hold this information is one way administrators may preserve this information. From a programming
perspective, it is important that residual scan and schema files that are uploaded from the client are removed once
the operation is complete. Both of these files will need to be removed from the client by the agent once data
processing is complete.

Since the program will be designed as a service that operates in the Windows “Services and Applications”
group. The potential exists for the service to be terminated on the client. In order to mitigate against an adversary
shutting off the service all together, appropriate provisions should be made on the local machine or through the use
of group policy in Active Directory to ensure that only enhanced accounts can modify and change the service.
Access Control Lists (ACLs) for the service can be created and applied to the group policy object. In the ACL,
specific users and groups can be added who have authority to start, stop, or pause the service [11].

Lastly, write access to the central database is given to clients with the agent deployed. While this is
necessary for the program to function, it may also represent a potential security vulnerability. There are two ways
authentication can be handled to the database server, through a hardcoded connection string or by using integrated
domain authentication. The most appropriate method largely depends on the environment where the solution will be
implemented.

The functionality of the program as well as the protection of aggregate data integrity were two key items
that needed to be considered from a security perspective. The writer was able to leverage the use of Active Directory
in many of the potential risks identified as a means to facilitate universal policy deployment creation and
implementation. This is one of the key advantages toward choosing to use Microsoft’s line of products for
development. More security precautions can be implemented since the program operates on a single vendor OS
which may be integrated into a single unified management structure.
Results

The results section speaks to the final versions of the client agent and web portal. In the agent software section, the program flow is presented, along with hierarchical overview of the all the classes created. Afterwards, an overview of the references and dependencies the agent utilizes are given. The next section details how the sample analyst front-end was laid out, and provides a basic template of how the aggregate data may be displayed in real time to the analyst, as well as how analysts may query the database directly. In the final section, a discussion regarding the schema definition file and a sample system snapshot is given.

Agent Software

The final version of the agent software is named CAITO (Collector of Actionable Intelligence for Threat Observations). When compiled as an MSI, the software is approximately 548Kb, which is relatively lightweight and an appropriate size for enterprise deployment. There are no advanced installation options that would complicate the automation of deployment; all required libraries are compiled with the MSI package.

Program Flow

The overall flow of the CAITO agent begins once the service is started in Windows. When invoked, the software collects two important attributes that are stored in the registry: the file name of the instruction file, and the IP address of the web server hosting the instruction file. Once the attributes are retrieved, an attempt is made to access the hosted instruction file. If this attempt is successful, then the XML elements in the instruction file are parsed for further processing by the agent, if not then the client will attempt to require registry attributes and try again.

The instruction file is parsed for the following information; the time to report, the database location, the database name, and subsequent directories to aggregate. Once these attributes are parsed, a check is done on the reporting time attribute, if the time on the client matches the reporting time given by the instruction file a random sleep is initiated before further progress is made in the process tree. If the agent time is not a match with the reporting time derived from the instruction file, then the agent sleeps for one minute before it accesses the instruction file again. The process repeats itself until the agent’s time and the reporting time finally match. This methodology facilitates daily reporting. If administrators desire a different type of reporting time frame, such as
every other day, or weekly, then adjustments would need to be made in the requisite class. Moving forward however, this functionality is something future versions of the agent could support in a more flexible manor.

Once the reporting time is reached and the random back off sequence is complete, the agent creates an XML file on the local client where system data will be aggregated. After this is done, client information is aggregated and written to the system XML file. Directory information is then enumerated next, and each directory is processed once at a time. Files within the given directory are processed and checked for presence of Alternate Data Streams (ADSs). If one or more streams are detected, then they are enumerated as well and written to the system state file.

Once files in a given directory are processed, then the parent directory is checked for presence of subdirectories. If there are subdirectories, then they themselves are enumerated along with any files and ADSs in a recursive fashion until the end of the directory tree is reached. After the parent directory given by the instruction file has all requisite information aggregated, the next directory in the instruction file is processed. If there are no more directories to process, the next phase of aggregation begins.

Network information is aggregated from the client next. All active and listening TCP connection are enumerated, and stored. Next, any listening UDP connections are listed and stored in the system state XML file. The next element in the flow is the enumeration of processes.

Each process in the process tree is enumerated individually. A given process has the information pertaining to it aggregated, and any subsequent threads running under that process are aggregated as well. Once all processes and their respective threads have been examined by the CAITO agent, the aggregation phase of runtime has completed.

The system state configuration file is completed and the last XML element is written to disk. The schema definition file is then pulled down from its remote server to the local client. If this operation completes successfully, then the upload sequence is initiated, and the agent passes the system state file along with the schema definition file to the database. Failure to retrieve the scheme definition file from the host will result in a reattempt from the agent. After the upload is complete, the schema and system state files are deleted from the client, and the overall process is reinitiated. Figure 17 represents a visual overview of the CAITO agent’s overall flow.
Class Overview

The classes developed for the agent were done so in a hierarchal fashion. There are four main levels of the hierarchy. Each level denotes a phase of the agent run time process. Figure 18 illustrates how all the classes are represented in their hierarchy.
The first level encompasses the main classes that are used to install the CAITO program as a service in the Windows OS, as well as instruction and details pertaining to the service itself. WindowsServiceInstaller.cs provides important installation details such as; the account for the system to install under, the service description and name as observed in “Services and Applications,” and the automatic start type for the service once installed. The CAITO.cs class provides functionality to the start and stop methods that the service uses.

Level two refers to the base class, Client.cs. The majority of agent runtime is performed in this class, and it is invoked by CAITO.cs. The Client.cs class handles procedure calls to the DataConnection.cs class and RegQuery.cs class where attributes are retrieved and stored either locally or remotely. The Client.cs class also handles the call to the Builder.cs class once the correct reporting time is achieved.

The third level contains the three classes that are invoked in by Client.cs. The DataConnection.cs class handles the processing of the remotely hosted instruction file and the retrieval of the schema definition file. The class also handles the upload of the system state file to the relational database. RegQuery.cs retrieves the local registry attributes the program needs to locate the instruction file. Builder.cs handles the creation of the system state.

Figure 18 – Hierarchal Overview of CAITO Classes
file, enumeration of directories, and all calls to subsequent classes responsible for the aggregation client state information.

The final level of the program hierarchy deals with the aggregation classes. FileAggregation.cs contains all methods responsible for returning file metadata attributes, such as MD5 and ADS data. NetworkAggregation.cs manages the functions which gather data on active and listening TCP connections currently active by the client as well as UDP listening connections. Finally, ProcessAggregation.cs enumerates data associated with processes running on the client; subsequent calls are made to ThreadAggregation.cs from Builder.cs for all threads belonging to a given process.

Dependencies

Each class represented in the CAITO agent hierarchy maintains a list of references to various components of the .NET Framework. There are also other Dynamic Link Libraries (DLLs) that are used by the agent which are not a direct part of .NET and are therefore packaged with the MSI as dependencies. They are as follows: Trinet.Core.IO.Ntfs.dll, sqxml4m.dll, and xblkld4r.dll.

The Trinet.Core.IO.Ntfs.dll assembly library was written by Richard Deeming, and was recently rewritten to support the release of .NET v3.5. According to the developer of the library, the .NET Framework does not support the native retrieval of ADSs associated with files. The following explanation elaborates further on this [12].

"If you attempt to open a FileStream on an alternative stream, you will get a ‘Path Format not supported’ exception. I have been unable to find any class in the CLR that provides support for alternative data streams...”

The addition of the author’s library makes it possible to easily detect and aggregate information pertaining to the existence of an ADS. The DLL is a third party extension, and compiled into the CAITO agent MSI.

The two remaining DLLs sqxml4m.dll and xblkld4r.dll are not part of the .NET framework, but are necessary because they both support the use of the XML Bulk Upload procedure used to store data from the client to the database. These libraries were to be retrieved from Microsoft SQL Server, and were registered using the regsvr32 utility. The xblkld4r and sqxml4m libraries are dependencies of the main library needed to execute the bulk upload, which is called xblkld4.dll. The library xblkld4.dll is added as a reference in the overall program, but since it depends on both xblkld4r and sqxml4m, it was necessary to add them as a dependency.
Finally, the last dependency for the program to run is version 3.5 or greater of the .NET Framework. The framework provides all the necessary assembly references and namespaces which are required at run time when the CAITO agent is invoked. Many other software applications use this framework, and it a standard installation found on Windows based operating systems.
Web Portal

The writer designed an accompanying web portal because he wanted to provide a comprehensive proof of concept, which details how an architecture supporting a forensically principled database schema could be implemented, and how the aggregate data could be queried and processed by an analyst. Doing so also partially illustrates the power behind the relational aspect of the database by developing graphs that display architecture metrics and lending some querying capabilities to the analyst. The web functionality and layout presented is only a basic template of how this data could be represented.

Functionality

By leveraging SqlDataSource objects in the ASP.NET language, the writer embedded SQL query results in several portions of the template web page. The embedded SQL queries display total amounts of snapshot stored in the database, total amount of ADS detections, total amount of reporting clients, and the data of the last report. The development of graphs was done using data populated from the backend database. Charts detailing the most popular connections and the number of reports in the last 24 hours can be observed.

Additionally, analysts are afforded the ability to query the database directly on the portal page. By typing in a keyword identifier into the search box, and selecting the table they would like to query, they are presented with the results. Query results are displayed using the GridView object provided in ASP.NET once the analyst submits their query.

By providing this kind of example functionality, the value of storing this information in a relational database is further realized. Since data is dynamically structured and mapped, analysts are able to draw correlations to clients possessing a particular indicator much quicker. The potential for data visualization and analysis enables both information security and operations staff to interpret respective metrics more effectively.

Figure 19 presents an illustration of the sample portal that an analyst might use. Embedded queries are in the top and middle sections of the web design, while the analyst query inputs are accepted below. Figure 20 represents a sample of the output of an analyst query using the GridView object in ASP.NET. Figure 21 shows another sample query result when the process name is searched.
Client Aggregation Command Center

Summary:

Reporting Clients: 1
ADS Count: 7
Snapshot Count: 7
Last Check-In: 10/19/2010 5:06:04 PM

Popular Connections

- 72.14.204.17
- 192.168.1.6
- 205.234.175.175
- 91.193.212.171
- 192.168.1.114

Reports In Last 12hrs

Figure 19 – Web Based Analyst Interface
Figure 20 – Sample IP Address Query Results from Portal

Figure 21 – Sample Process Query Results from Portal
Conclusions

A forensically principled client indicator database provides analysts with the proper amount of actionable intelligence they need in order to rapidly identify compromises within their network and prevent the exposure of sensitive information. The data pivoted on may lead to the discovery of other indicators which assist in building a comprehensive list of incident related indicators to mitigate on. The solution also assists analysts in assessing the feasibility of deploying host-based mitigation strategies, by providing an instant snapshot of the impact it would have across clients in the enterprise.

The threat facing critical computing infrastructure has shifted, adversaries are able to penetrate networks by leveraging a unique set of tools that are not detected by common security tools until it is too late. The information security community must adopt an intelligence based approach toward the remediation of computer incidents. Enterprise intelligence gathering begins at the client level, because that is where the adversaries reside.
Future Work

A solution that provides actionable intelligence on client nodes has been provided in this document. There are a multitude of other areas from which research can be conducted to improve and enhance the solutions capabilities. Future work concerning this design should concentrate in the areas of increased security, SIM integration, correlative analysis, performance monitoring, and enhanced portal design.

Security

While adversaries may be limited in their ability to locate the host for the instruction file provided to the agent through employing appropriate ACLs within the registry, the file itself is still publicly viewable. There are several mechanisms that may be implemented at the operations level to help safeguard against this. The use of Integrated Windows Authentication for web servers is one way. However, there are programmatic solutions that may offer a stronger safeguard.

The use of authentication tokens might be one way to ensure the configuration file is accessed only by agents. CAITO agents running on clients could use an algorithm to generate an authentication token, and use this to provide to the web server to access the instruction file. Alternatively, encoding the contents of the instruction file using a specific key may also offer a layer of security by obfuscating the publicly viewable information. Implementation of one or both of these suggestions would help to decrease the chance of an intelligence gain by the adversary.

Integration

The relational schema generated has been presented as an independent database. There is no reason however, that the aggregate data and schema definition could not be tied into a larger SIM dataset. In this fashion, data generated using CAITO agents would exist alongside other elements of SIM data, such as proxy, firewall, and IDS logs. The ability to integrate this dataset into existing SIM platforms may be a highly valued capability by organizations seeking to centralize all sources of security intelligence.

Correlative Analysis

The data set that is generated through the deployment of CAITO agents presents the opportunity for deep data mining and correlative analysis. One idea to automate the analysis of this vast dataset would be to develop a
background daemon which is designed to trigger on certain pre-defined indicators that warrant further investigation by an analyst.

Such a pre-defined indicator might be the dropping of known utilities onto the file system such as password crackers, or other well-known tools attackers might use to further expand their control in the network. The presence of these tools may warrant one level of alert generation, while the location of these tools might trigger a more critical alert; when a malicious tool is found in a popular hiding place for attackers such as C:\Recycler for example. Many more mappings of potentially suspicious activity can be made with the aggregate data set.

Correlative analysis may be further augmented by integrating a pre-existing database of more detailed, client based indicators. These may be attributes such as MD5 hashes, IP addresses, or files written to a certain directory. Data sets derived from past incidents or other means, can be tied into aggregate information generated from CAITO and alerted upon in an automated fashion.

**Benchmark Statistics**

Operational staff would probably appreciate more detailed research into the impact the implementation of this reporting structure would have on their environment. In particular, a review of metrics as they apply to the corporate network would be helpful for operations staff to gauge the impact. The bandwidth impact of clients reporting with varying data sets in certain environments might be helpful.

Metrics relating to system performance would also be of value. For the client, determining average CPU time and load when a scan is kicked off on baseline PCs helps assess impact to the client clearly. Also, for the servers hosting the instruction and schema definition files, a study should be done assessing the performance impact of many clients accessing and retrieving information from these hosts. The results may or may not necessitate changes in how back off sequences are maintained.

The database server should also be benchmarked, and studies should be done pertaining to performance impact of larger data sets. Ultimately, the question of how many clients one aggregation node can support over time should be answered. It would also be interesting to know how many snapshots a database server with a given amount of hardware can support without significant performance impact. Results of these studies could be used to try to improve the overall schema design.
By making these kinds of observations, operations staff will be able to make a better determination of the impact deployment of this architecture will have on various elements of their infrastructure. It is difficult to state metrics for networks, servers, and workstations given how technology has rapidly advanced over time. Measurements using a pre-determined baseline can help others estimate and better understand issues surrounding implementation.

**Portal Design**

Concerning the web portal, there are many improvements that could be made with its design. A sample was provided to illustrate the potential for a front-end. Utilization of Web2.0 technologies such as Flash and AJAX would further enhance its usability.

The ability to view clients as objects in a web browser, and instantaneously view quick metrics about them would be beneficial to analysts. Dragging client objects to different pools of interest is a web feature analysts might also find useful during an investigation. The ability to save these interest pools and collaborate with other analyst about them using the portal interface would facilitate a unified response process.
Appendices

Instructions

Example InfoFile.xml

```xml
<ROOT>
    <Time>12:10</Time>
    <DBServer>192.168.1.8</DBServer>
    <DBName>SIMS</DBName>
    <Directory>C:\WINDOWS\Temp</Directory>
    <Directory>C:\Windows\Web</Directory>
    <Directory>C:\Users\Jason\Desktop\testdir</Directory>
    <Directory>C:\Windows\Prefetch</Directory>
</ROOT>
```
Example Client Sys.xml File

```xml
<?xml version="1.0"?>
<Snapshots SnapDate="10/19/2010 5:32:11 PM" ClientMAC="002100E79D4D"
IPAddress="192.168.1.114" Hostname="Livewire">
  <Directories DirectoryName="C:\WINDOWS\Temp">
    <Files FileName="coinlog.log" LastWriteTime="10/18/2010 8:05:23 PM"
FileSize="4832" FileMD5="910d821efc6b36466f05460b4563219" FileADS="0" />
    <Files FileName="dd_clwireg.txt" LastWriteTime="10/10/2010 4:46:50 PM"
LastAccessTime="10/10/2010 4:46:49 PM" FileSize="4153"
FileSize="2e2d77b17daa81f0356f0c0f945682ab5" FileADS="0" />
    <Files FileName="fwtsqmfile00.sqm" LastWriteTime="4/16/2010 8:09:24 PM"
FileSize="632" FileMD5="967d8fafl1fc7770c5f34457f841feaae" FileADS="0" />
    <Files FileName="fwtsqmfile01.sqm" LastWriteTime="4/24/2010 4:28:21 PM"
FileSize="632" FileMD5="485ad1bedacc26d5d5eb2465dc3e286c" FileADS="0" />
    <Files FileName="fwtsqmfile02.sqm" LastWriteTime="5/1/2010 4:38:34 PM"
FileSize="120" FileMD5="935ff0f82b77c22304c95c3f535db90c" FileADS="0" />
    <Files FileName="fwtsqmfile03.sqm" LastWriteTime="5/11/2010 12:36:08 PM"
FileSize="632" FileMD5="ab31a1030ae50f0f9ff14fd20dafed8d1" FileADS="0" />
    <Files FileName="fwtsqmfile04.sqm" LastWriteTime="5/22/2010 12:45:32 AM"
FileSize="632" FileMD5="4a87a006285a89a1b7dd86252d401ac" FileADS="0" />
    <Files FileName="fwtsqmfile05.sqm" LastWriteTime="6/2/2010 11:35:09 AM"
FileSize="120" FileMD5="05564a50115b0aa0d5f65c0f90248a" FileADS="0" />
    <Files FileName="fwtsqmfile06.sqm" LastWriteTime="6/13/2010 12:13:52 PM"
FileSize="120" FileMD5="b87a930d3b1af368ea4f0b0aa32f4364" FileADS="0" />
    <Files FileName="fwtsqmfile07.sqm" LastWriteTime="6/23/2010 12:22:59 PM"
FileSize="632" FileMD5="b024a3e5857f230bc26d1d67447790d" FileADS="0" />
    <Files FileName="fwtsqmfile08.sqm" LastWriteTime="7/17/2010 6:59:11 PM"
FileSize="632" FileMD5="e42abf66d87bb197c9bc5930b3cad9" FileADS="0" />
  </Directories>
</Snapshots>
```
<Files FileName="fwtsqmfile09.sqm" LastWriteTime="8/15/2010 8:49:19 PM" LastAccessTime="8/15/2010 8:49:19 PM" FileCreationTime="8/15/2010 8:49:19 PM" FileSize="120" FileMD5="316f743bd8476e53e6bc5d69d7a4ab66" FileADS="0" />
<Files FileName="fwtsqmfile13.sqm" LastWriteTime="10/16/2010 5:30:02 PM" LastAccessTime="10/16/2010 5:30:02 PM" FileCreationTime="10/16/2010 5:30:02 PM" FileSize="120" FileMD5="21a91012da16de142bf8e0af919b61ff" FileADS="0" />
<Files FileName="MpCmdRun.log" LastWriteTime="10/19/2010 3:36:19 PM" LastAccessTime="10/16/2010 12:02:26 AM" FileCreationTime="10/16/2010 12:02:26 AM" FileSize="26668" FileMD5="ef51909b57f7d88a969a03991d81c38" FileADS="0" />
</Files>
<Directories DirectoryName="C:\WINDOWS\Temp\{4d36e96d-e325-11ce-bfcl-08002be10318}0000" />
<Directories DirectoryName="C:\WINDOWS\Temp\RtSigs\Data" />
<Directories DirectoryName="C:\WINDOWS\Temp\History\Results">

<File FileName="Text.txt" LastWriteTime=\"8/15/2010 11:33:48 PM\" LastAccessTime=\"8/15/2010 11:33:48 PM\" FileSize="0" FileMD5="d41d8cd98f00b204e9800998ecf8427e" FileADS="0" />
</Directories>

<Directories DirectoryName="C:\Windows\Web" />
<Directories DirectoryName="C:\Windows\Web\Wallpaper">

</Directories>

<Directories DirectoryName="C:\Windows\Web\Wallpaper\Wallpaper">

</Directories>

<Directories DirectoryName="C:\Windows\Web\Wallpaper\Wallpaper\Data">

</Directories>

<Directories DirectoryName="C:\Windows\Web\Wallpaper\Wallpaper\Data\Swoosh.jpg">

</Directories>

<Directories DirectoryName="C:\Windows\Web\Wallpaper\Wallpaper\Data\Swoosh.jpg">

<File FileName="1280x1024Lpeacock.jpg" LastWriteTime=\"11/14/2006 3:35:34 AM\" LastAccessTime=\"6/15/2009 7:31:52 AM\" FileCreationTime=\"6/15/2009 7:31:52 AM\"FileSize="528776" FileMD5="6d82bbebf142e71d2be872f24791742603a6c3956b3828a2e57423" FileADS="0" />
</Directories>

<Directories DirectoryName="C:\Windows\Web\Wallpaper\Wallpaper\Data\Swoosh.jpg">

<File FileName="1280x800Lpeacock.jpg" LastWriteTime=\"11/14/2006 3:35:18 AM\" LastAccessTime=\"6/15/2009 7:31:52 AM\" FileCreationTime=\"6/15/2009 7:31:52 AM\"FileSize="668054" FileMD5="6c361bee29a08ae0ca218e41204ecf4a4" FileADS="0" />
</Directories>

<Directories DirectoryName="C:\Windows\Web\Wallpaper\Wallpaper\Data\Swoosh.jpg">

</Directories>

<Directories DirectoryName="C:\Windows\Web\Wallpaper\Wallpaper\Data\Swoosh.jpg">

</Directories>

<Directories DirectoryName="C:\Windows\Web\Wallpaper\Wallpaper\Data\Swoosh.jpg">

</Directories>

</Directories>
<Files FileADS="" />


46
<Files FileName="img11.jpg" LastWriteTime="11/2/2006 12:35:26 PM" LastAccessTime="11/2/2006 12:35:26 PM" FileCreationTime="11/2/2006 12:35:26 PM" FileSize="1522516" FileMD5="612df35904ce0a87d1c3349683a01" FileADS="0" />
<Files FileName="img19.jpg" LastWriteTime="11/2/2006 12:35:25 PM"


<Files FileName="img32.jpg" LastWriteTime="11/2/2006 12:35:26 PM"
LastAccessTime="11/2/2006 12:35:26 PM"
FileCreationTime="11/2/2006 12:35:26 PM"
FileSize="1375915" FileMD5="clb53eba51f7be45c8e4fd77f15197dce" FileADS="0">
<Files FileName="img33.jpg" LastWriteTime="11/2/2006 12:35:26 PM"
LastAccessTime="11/2/2006 12:35:26 PM"
FileCreationTime="11/2/2006 12:35:26 PM"
FileSize="1511911" FileMD5="d249b12aefb43c8fab53042f8c086850" FileADS="0">
<Files FileName="img34.jpg" LastWriteTime="11/2/2006 12:35:26 PM"
LastAccessTime="11/2/2006 12:35:26 PM"
FileCreationTime="11/2/2006 12:35:26 PM"
FileSize="1532026" FileMD5="adda19a160f05ef46c9306e15223782" FileADS="0">
<Files FileName="img35.jpg" LastWriteTime="11/2/2006 12:35:26 PM"
LastAccessTime="11/2/2006 12:35:26 PM"
FileCreationTime="11/2/2006 12:35:26 PM"
FileSize="717578" FileMD5="d136b7973136f207d62cb83267488127" FileADS="0">
<Files FileName="img36.jpg" LastWriteTime="11/2/2006 12:35:25 PM"
LastAccessTime="11/2/2006 12:35:25 PM"
FileCreationTime="11/2/2006 12:35:25 PM"
FileSize="1557486" FileMD5="860b477aee3538917f67bf9534f6738b" FileADS="0">
<Files FileName="img4.jpg" LastWriteTime="11/2/2006 12:35:27 PM"
LastAccessTime="11/2/2006 12:35:27 PM"
FileCreationTime="11/2/2006 12:35:27 PM"
FileSize="1487550" FileMD5="715643a08f5bdd8d99dcb8847ca9e" FileADS="0">
<Files FileName="img5.jpg" LastWriteTime="11/2/2006 12:35:27 PM"
LastAccessTime="11/2/2006 12:35:27 PM"
FileCreationTime="11/2/2006 12:35:27 PM"
FileSize="1237529" FileMD5="1f48d5e28f08eca651a6cddeb8a7fe5b" FileADS="0">
<Files FileName="img6.jpg" LastWriteTime="11/2/2006 12:35:27 PM"
LastAccessTime="11/2/2006 12:35:27 PM"
FileCreationTime="11/2/2006 12:35:27 PM"
FileSize="1603603" FileMD5="af12c05a6cede03a71365de6192def0c" FileADS="0">
<Files FileName="img7.jpg" LastWriteTime="11/2/2006 12:35:27 PM"
LastAccessTime="11/2/2006 12:35:27 PM"
FileCreationTime="11/2/2006 12:35:27 PM"
FileSize="1279829" FileMD5="d88900528a3e5e3f43d63808c8baa1ee" FileADS="0">
<Files FileName="img8.jpg" LastWriteTime="11/2/2006 12:35:27 PM"
LastAccessTime="11/2/2006 12:35:27 PM"
FileCreationTime="11/2/2006 12:35:27 PM"
FileSize="1477946" FileMD5="29c591bfa7c165e2c0a5069f9227101" FileADS="0">
</Files>
</Files>
</Files>
</Files>
</Files>
</Files>
</Files>
</Files>
<Files FileName="mybat.bat" LastWriteTime="11/2/2006 12:35:26 PM"
LastAccessTime="11/2/2006 12:35:26 PM"
FileCreationTime="11/2/2006 12:35:26 PM"
FileSize="67" FileMD5="057cadd4bbda746bd7c7bd277f14126" FileADS="0">
<Files FileName="myfile.txt" LastWriteTime="11/2/2006 12:35:27 PM"
LastAccessTime="11/2/2006 12:35:26 PM"
FileCreationTime="11/2/2006 12:35:26 PM"
FileSize="22" FileMD5="c33df5a150db65b0ab546cfba396b" FileADS="1">
<AdsData ADSSize="19" ADSSPath="C:\Users\Jason\Desktop\testdir\myfile.txt:secret.txt"
</Files>

</Directories>

<Directories DirectoryName="C:\Users\Jason\Desktop\testdir\newdir">
</Files>

<Directories DirectoryName="C:\Windows\Prefetch">
<Files FileName="AgAppLaunch.db" LastWriteTime="7/31/2009 1:30:15 PM" LastAccessTime="7/31/2009 1:30:15 PM" FileCreationTime="7/31/2009 1:30:15 PM" FileSize="332116" FileMD5="769a2eabde49083ff4a67785ba29a7ca" FileADS="0" />
<Files FileName="AgCx_S1_S-1-5-21-4019528468-2161341046-492441667-1003.snp.db" LastWriteTime="4/17/2010 7:01:49 PM" LastAccessTime="4/17/2010 7:01:49 PM" FileCreationTime="4/17/2010 7:01:49 PM" FileSize="3156206" FileMD5="cbcd1eb1699e138cd8f8db4d4555c70e9" FileADS="0" />
<Files FileName="AgCx_SC1.db" LastWriteTime="6/24/2010 1:15:21 AM" LastAccessTime="8/1/2009 11:28:01 AM" FileCreationTime="8/1/2009 11:28:01 AM" FileSize="720925" FileMD5="465bd8b3824c04e8bfa81a778c0b2e1" FileADS="0" />
<Files FileName="AgGlFgAppHistory.db" LastWriteTime="10/19/2010 8:35:57 PM" LastAccessTime="8/1/2009 11:27:53 AM" FileCreationTime="8/1/2009 11:27:53 AM" FileSize="1425350" FileMD5="6bf9a3c90a3f6878ceff315d2d9c80a" FileADS="0" />
</Files>
</Directories>
<Files FileMD5="09d1ed2d98bf2380e3e97216831c8faa" FileADS="0" />
  <Files FileName="MMC.EXE-D557C836.pf" LastWriteTime="10/18/2010 9:41:07 PM" LastAccessTime="10/18/2010 9:41:07 PM" FileCreationTime="10/18/2010 9:41:07 PM" FileSize="50606" FileMD5="89a64e46cb191f7c467dfc5affeb2a" FileADS="0" />
  <Files FileName="MPAS-FE_BD.EXE-BB62FD5E.pf" LastWriteTime="10/19/2010 3:34:40 PM" LastAccessTime="10/19/2010 3:34:40 PM" FileCreationTime="10/19/2010 3:34:40 PM" FileSize="11456" FileMD5="9d00dcb64363448c99a39a74cee305e" FileADS="0" />
  <Files FileName="MPCMDRUN.EXE-F401FB4.pf" LastWriteTime="10/19/2010 3:36:29 PM" LastAccessTime="4/30/2010 9:32:01 PM" FileCreationTime="4/30/2010 9:32:01 PM" FileSize="6082" FileMD5="ddff8b13d966c556ca14d42e7ccfc" FileADS="0" />
  <Files FileName="MPSIGSTUB.EXE-D3661409.pf" LastWriteTime="10/19/2010 3:34:59 PM" LastAccessTime="10/19/2010 3:34:59 PM" FileCreationTime="10/19/2010 3:34:59 PM" FileSize="17464" FileMD5="e123adca932a04aef2515c3474559" FileADS="0" />
  <Files FileName="MSACCESS.EXE-5B4ABA51.pf" LastWriteTime="10/18/2010 9:03:23 PM" LastAccessTime="10/18/2010 9:03:23 PM" FileCreationTime="10/18/2010 9:03:23 PM" FileSize="55500" FileMD5="5d895f0e20ebe8946250e9c8229cd3" FileADS="0" />
  <Files FileName="MSIKEEXEC.EXE-A2D5CB6.pf" LastWriteTime="10/19/2010 9:11:24 PM" LastAccessTime="8/15/2010 8:35:19 PM" FileCreationTime="8/15/2010 8:35:19 PM" FileSize="66300" FileMD5="cee43cd6a2d5b8badcd8ec3b6e03cc582" FileADS="0" />
  <Files FileName="MSPAINT.EXE-76E10824.pf" LastWriteTime="10/19/2010 9:07:45 PM" LastAccessTime="10/18/2010 1:13:40 AM" FileCreationTime="10/18/2010 1:13:40 AM" FileSize="43590" FileMD5="66976c79526371d7155d52cf02c7dd36" FileADS="0" />
  <Files FileName="MSTSC.EXE-76A46E8A.pf" LastWriteTime="10/19/2010 8:25:18 PM" LastAccessTime="10/18/2010 6:08:18 PM" FileCreationTime="10/18/2010 6:08:18 PM" FileSize="55804" FileMD5="058bad7b2a3a1f1894917fa8fe30cdd" FileADS="0" />
  <Files FileName="ORCA.EXE-C834FA8.pf" LastWriteTime="10/18/2010 1:12:15 AM" LastAccessTime="10/18/2010 1:12:15 AM" FileCreationTime="10/18/2010 1:12:15 AM" FileSize="52538" FileMD5="e89d8b86f84ad32056b31a99501d6c" FileADS="0" />
</Files>


<Files FileName="REG.EXE-E7EBBD26.pf" LastWriteTime="10/19/2010 9:28:47 PM" LastAccessTime="10/18/2010 4:54:51 PM" FileCreationTime="10/18/2010 4:54:51 PM" FileSize="11064" FileMD5="48407b8a8c2264304e3f4767ad89e2c0" FileADS="0"/>

<Files FileName="REGEDIT.EXE-90FEEA06.pf" LastWriteTime="10/19/2010 8:31:45 PM" LastAccessTime="10/18/2010 4:56:37 PM" FileCreationTime="10/18/2010 4:56:37 PM" FileSize="24314" FileMD5="eb66790aeb0344a8699dea015f0599d7" FileADS="0"/>

<Files FileName="REGSVR32.EXE-8461DBEE.pf" LastWriteTime="10/19/2010 9:29:19 PM" LastAccessTime="10/18/2010 7:02:03 PM" FileCreationTime="10/18/2010 7:02:03 PM" FileSize="27676" FileMD5="3240dcf89f75262f13f3b2d1e5868445d" FileADS="0"/>


<Files FileName="SEARCHPROTOCOLHOST.EXE-0CB6CADE.pf" LastWriteTime="10/19/2010 9:28:21 PM" LastAccessTime="7/31/2009 1:31:50 PM" FileCreationTime="7/31/2009 1:31:50 PM" FileSize="23506" FileMD5="fbd02d51bc005e25527306e44fc5c49d" FileADS="0"/>


<Files FileName="SIMS_CLIENT.VSHOST.EXE-91E3B8D7.pf" LastWriteTime="10/19/2010 9:29:56 PM" LastAccessTime="10/18/2010 8:33:05 PM" FileCreationTime="10/18/2010 8:33:05 PM" FileSize="113086" FileMD5="0ebf2b2ffcc2201111a42d7b6198e328d" FileADS="0"/>

<Files FileName="SNVOL.EXE-5D4CC7D6.pf" LastWriteTime="10/18/2010 2:00:02 AM" LastAccessTime="10/18/2010 2:00:02 AM" FileCreationTime="10/18/2010 2:00:02 AM" FileSize="23074" FileMD5="d531f388410f185046d420efaab7757" FileADS="0"/>

<Files FileName="SOFTWAREUPDATE.EXE-631B74E4.pf" LastWriteTime="10/18/2010 4:16:31 PM" LastAccessTime="10/18/2010 4:16:31 PM" FileCreationTime="10/18/2010 4:16:31 PM" FileSize="2630" FileMD5="84f0e68a7a1de96120f7db39a90fbb61" FileADS="0"/>

<Files FileName="SQLWB.EXE-COF2BFCC0.pf" LastWriteTime="10/19/2010 7:11:20 PM" LastAccessTime="10/19/2010 7:11:20 PM" FileCreationTime="10/19/2010 7:11:20 PM" FileSize="23074" FileMD5="8a8e3c8148921bc5ad490d16abe27a8" FileADS="0"/>

<Files FileName="SVCHOST.EXE-7CFEDEA3.pf" LastWriteTime="10/19/2010 7:11:20 PM" LastAccessTime="10/19/2010 7:11:20 PM" FileCreationTime="10/19/2010 7:11:20 PM" FileSize="23074" FileMD5="8a8e3c8148921bc5ad490d16abe27a8" FileADS="0"/>
FileCreationTime="10/19/2010 1:30:55 PM" FileAttributes="0" FileMD5="5016c83dd09267d5dd54a92324d9440d" FileADS="0" />
<Files FileName="TASKENG.EXE-48D4E289.pf"
FileCreationTime="7/31/2009 1:35:59 PM" FileAttributes="0" FileMD5="5f4819c35327fbba227f2349882b2689" FileADS="0" />
<Files FileName="TASMGR.EXE-5F5F473D.pf"
LastWriteTime="10/18/2010 8:40:30 PM" LastAccessTime="10/18/2010 7:38:22 PM"
FileCreationTime="10/18/2010 7:38:22 PM" FileAttributes="0" FileMD5="db6f750e22a3d4f47b6a595aaa60a665" FileADS="0" />
<Files FileName="TRUSTEDINSTALLER.EXE-3CC531E5.pf"
LastWriteTime="10/19/2010 3:32:05 PM" LastAccessTime="8/2/2009 2:30:19 PM"
FileCreationTime="8/2/2009 2:30:19 PM" FileAttributes="0" FileMD5="53484" FileADS="0" />
<Files FileName="VERCLSID.EXE-7CS2E31C.pf"
FileCreationTime="7/31/2009 1:35:51 PM" FileAttributes="0" FileMD5="9ff4642b46212e1604654999775fe7652" FileADS="0" />
<Files FileName="WISIO.EXE-60D85FC1.pf"
FileCreationTime="10/18/2010 10:32:27 PM" FileAttributes="0" FileMD5="112930" FileADS="0" />
<Files FileName="VSSVC.EXE-BBAFC319.pf"
LastWriteTime="10/19/2010 9:11:24 PM" LastAccessTime="7/31/2009 1:30:52 PM"
FileCreationTime="7/31/2009 1:30:52 PM" FileAttributes="0" FileMD5="32392" FileADS="0" />
<Files FileName="W3WP.EXE-53273A08.pf"
LastWriteTime="10/19/2010 8:32:28 PM" LastAccessTime="10/18/2010 6:53:49 PM"
FileCreationTime="10/18/2010 6:53:49 PM" FileAttributes="0" FileMD5="41636" FileADS="0" />
<Files FileName="WEBDEV.WEBSERVER.EXE-9C4082A8.pf"
LastWriteTime="10/19/2010 8:27:48 PM" LastAccessTime="10/19/2010 8:27:48 PM"
FileCreationTime="10/19/2010 8:27:48 PM" FileAttributes="0" FileMD5="6392" FileADS="0" />
<Files FileName="WERCON.EXE-E36BD04E.pf"
LastWriteTime="10/19/2010 4:08:56 AM" LastAccessTime="8/18/2010 3:05:47 AM"
FileCreationTime="8/18/2010 3:05:47 AM" FileAttributes="0" FileMD5="117290" FileADS="0" />
<Files FileName="WERMGR.EXE-0F2AC89C.pf"
FileCreationTime="6/13/2010 12:02:18 PM" FileAttributes="0" FileMD5="23682" FileADS="0" />
<Files FileName="WINWORD.EXE-C91725A1.pf"
LastWriteTime="10/17/2010 3:00:06 PM" LastAccessTime="9/21/2010 12:05:08 PM"
FileCreationTime="9/21/2010 12:05:08 PM" FileAttributes="0" FileMD5="175880" FileADS="0" />
<Files FileName="WIRESHARK.EXE-76B1D076.pf"
LastWriteTime="10/18/2010 9:02:49 PM" LastAccessTime="10/18/2010 9:02:49 PM"
FileCreationTime="10/18/2010 9:02:49 PM" FileAttributes="0" FileMD5="163600" FileADS="0" />
<Files FileName="WMIPRVS.EXE-1628051C.pf"
LastWriteTime="10/19/2010 8:34:04 PM" LastAccessTime="7/31/2009 1:33:17 PM"
ActiveTcpConnections

RemoteTCPConnection="127.0.0.1:49157" TcpConnectionState="Established" /
ActiveTcpConnections LocalTCPConnection="127.0.0.1:27015"
RemoteTCPConnection="127.0.0.1:49181" TcpConnectionState="Established" /
ActiveTcpConnections LocalTCPConnection="127.0.0.1:49182"
RemoteTCPConnection="127.0.0.1:49180" TcpConnectionState="Established" /
ActiveTcpConnections LocalTCPConnection="127.0.0.1:49180"
RemoteTCPConnection="127.0.0.1:49179" TcpConnectionState="Established" /
ActiveTcpConnections LocalTCPConnection="127.0.0.1:49179"
RemoteTCPConnection="127.0.0.1:49178" TcpConnectionState="Established" /
ActiveTcpConnections LocalTCPConnection="127.0.0.1:49178"
RemoteTCPConnection="127.0.0.1:49177" TcpConnectionState="Established" /
ActiveTcpConnections LocalTCPConnection="127.0.0.1:49177"
RemoteTCPConnection="127.0.0.1:49176" TcpConnectionState="Established" /
ActiveTcpConnections LocalTCPConnection="127.0.0.1:49176"
RemoteTCPConnection="127.0.0.1:49175" TcpConnectionState="Established" /
ActiveTcpConnections LocalTCPConnection="127.0.0.1:49175"
RemoteTCPConnection="127.0.0.1:49174" TcpConnectionState="Established" /
ActiveTcpConnections LocalTCPConnection="127.0.0.1:49174"
RemoteTCPConnection="127.0.0.1:49173" TcpConnectionState="Established" /
ActiveTcpConnections LocalTCPConnection="127.0.0.1:49173"
RemoteTCPConnection="127.0.0.1:49172" TcpConnectionState="Established" /
ActiveTcpConnections LocalTCPConnection="127.0.0.1:49172"
RemoteTCPConnection="127.0.0.1:49171" TcpConnectionState="Established" /
ActiveTcpConnections LocalTCPConnection="127.0.0.1:49171"
RemoteTCPConnection="127.0.0.1:49170" TcpConnectionState="Established" /
ActiveTcpConnections LocalTCPConnection="127.0.0.1:49170"
RemoteTCPConnection="127.0.0.1:49169" TcpConnectionState="Established" /
ActiveTcpConnections LocalTCPConnection="127.0.0.1:49169"
RemoteTCPConnection="127.0.0.1:49168" TcpConnectionState="Established" /
ActiveTcpConnections LocalTCPConnection="127.0.0.1:49168"
RemoteTCPConnection="127.0.0.1:49167" TcpConnectionState="Established" /
ActiveTcpConnections LocalTCPConnection="127.0.0.1:49167"
<ActiveTcpConnections LocalTCPConnection="192.168.1.114:56517"
RemoteTCPConnection="91.199.212.171:80" TcpConnectionState="CloseWait"/>
<ActiveTcpConnections LocalTCPConnection="192.168.1.114:56518"
RemoteTCPConnection="208.116.56.69:80" TcpConnectionState="CloseWait"/>
<ActiveTcpConnections LocalTCPConnection="192.168.1.114:56601"
RemoteTCPConnection="72.14.204.83:443" TcpConnectionState="Established"/>
<ActiveTcpConnections LocalTCPConnection="192.168.1.114:56602"
RemoteTCPConnection="72.14.204.19:443" TcpConnectionState="Established"/>
<ActiveTcpConnections LocalTCPConnection="192.168.1.114:56603"
RemoteTCPConnection="65.55.12.249:80" TcpConnectionState="Established"/>
<ActiveTcpConnections LocalTCPConnection="192.168.1.114:56604"
RemoteTCPConnection="72.14.204.99:80" TcpConnectionState="Established"/>
<ActiveTcpConnections LocalTCPConnection="192.168.1.114:56605"
RemoteTCPConnection="72.14.204.99:80" TcpConnectionState="Established"/>
<ActiveTcpConnections LocalTCPConnection="192.168.1.114:56606"
RemoteTCPConnection="72.14.204.99:80" TcpConnectionState="Established"/>
<TcpListeningConnections TCPEndPointID="0.0.0.0:80" />
<TcpListeningConnections TCPEndPointID="0.0.0.0:135" />
<TcpListeningConnections TCPEndPointID="0.0.0.0:445" />
<TcpListeningConnections TCPEndPointID="0.0.0.0:5357" />
<TcpListeningConnections TCPEndPointID="0.0.0.0:49152" />
<TcpListeningConnections TCPEndPointID="0.0.0.0:49153" />
<TcpListeningConnections TCPEndPointID="0.0.0.0:49154" />
<TcpListeningConnections TCPEndPointID="0.0.0.0:49155" />
<TcpListeningConnections TCPEndPointID="0.0.0.0:49156" />
<TcpListeningConnections TCPEndPointID="127.0.0.1:5354" />
<TcpListeningConnections TCPEndPointID="127.0.0.1:27015" />
<TcpListeningConnections TCPEndPointID="192.168.1.114:139" />
<UdpListeningConnections UDPEndPointID="0.0.0.0:123" />
<UdpListeningConnections UDPEndPointID="0.0.0.0:500" />
<UdpListeningConnections UDPEndPointID="0.0.0.0:3702" />
<UdpListeningConnections UDPEndPointID="0.0.0.0:4900" />
<UdpListeningConnections UDPEndPointID="0.0.0.0:5355" />
<UdpListeningConnections UDPEndPointID="0.0.0.0:49152" />
<UdpListeningConnections UDPEndPointID="0.0.0.0:49545" />
<UdpListeningConnections UDPEndPointID="0.0.0.0:61880" />
<UdpListeningConnections UDPEndPointID="127.0.0.1:1900" />
<UdpListeningConnections UDPEndPointID="127.0.0.1:60791" />
<UdpListeningConnections UDPEndPointID="192.168.1.114:137" />
<UdpListeningConnections UDPEndPointID="192.168.1.114:138" />
<UdpListeningConnections UDPEndPointID="192.168.1.114:1900" />
<UdpListeningConnections UDPEndPointID="192.168.1.114:5353" />
<UdpListeningConnections UDPEndPointID="192.168.1.114:60790" />
</ProcessList PID="780" ProcessName="igfxpers"
ProcessOwner="Livewire\Jason" StartTime="10/17/2010 10:47:55 AM"
CPUTime="00:00:00.5460035" PhysicalMemSize="53972992" Threads="2" />
</ThreadList>
</ProcessList>
</ProcessList PID="5504" ProcessName="mspaint"
ProcessOwner="Livewire\Jason" StartTime="10/19/2010 5:07:35 PM"
CPUTime="00:00:05.8812377" PhysicalMemSize="104337408" Threads="4" />
</ThreadList>
</ProcessList>
</ProcessList>
</ProcessList PID="4796" StartTime="10/19/2010 5:07:35 PM"
CPUTime="00:00:04.2120270" Priority="9" StartAddress="2002869848" />
<ThreadList ThreadID="900" StartTime="10/19/2010 5:07:35 PM">
  <CPUTime time="00:00:00.0156001" />
  <CPUTime time="00:00:00.0780005" />
  <CPUTime time="00:00:00.4992032" />
  <CPUTime time="00:00:00.1092007" />
</ThreadList>

<ThreadList ThreadID="5680" StartTime="10/19/2010 5:07:36 PM">
  <CPUTime time="00:00:00" />
  <CPUTime time="00:00:00.0156001" />
  <CPUTime time="00:00:00.0780005" />
  <CPUTime time="00:00:00.4992032" />
</ThreadList>

<ThreadList ThreadID="5056" StartTime="10/19/2010 5:30:19 PM">
  <CPUTime time="00:00:00" />
  <CPUTime time="00:00:00.1092007" />
  <CPUTime time="00:00:00.0780005" />
  <CPUTime time="00:00:00.4992032" />
</ThreadList>

</ProcessList>

<ProcessList PID="4244" ProcessName="regedit">
  <CPUTime time="00:00:00.2.1216136" />
  <CPUTime time="00:00:00.02.1216136" />
  <CPUTime time="00:00:00.1560010" />
</ProcessList>

<ProcessList PID="1556" ProcessName="jusched">
  <CPUTime time="00:00:00.2.1216136" />
  <CPUTime time="00:00:00.02.1216136" />
  <CPUTime time="00:00:00.1560010" />
</ProcessList>

<ProcessList PID="4888" ProcessName="devenv">
  <CPUTime time="00:00:00.2.1216136" />
  <CPUTime time="00:00:00.02.1216136" />
  <CPUTime time="00:00:00.1560010" />
</ProcessList>

<ThreadList ThreadID="2232" StartTime="10/18/2010 6:31:42 PM" />
<ThreadList ThreadID="5344" StartTime="10/18/2010 6:31:45 PM" />
<ThreadList ThreadID="4980" StartTime="10/18/2010 6:31:45 PM" />
<ThreadList ThreadID="1676" StartTime="10/18/2010 6:31:46 PM" />
<ThreadList ThreadID="5408" StartTime="10/18/2010 6:31:59 PM" />
<ThreadList ThreadID="5244" StartTime="10/18/2010 6:32:07 PM" />
<ThreadList ThreadID="4052" StartTime="10/18/2010 6:32:17 PM" />
<ThreadList ThreadID="1324" StartTime="10/18/2010 6:32:20 PM" />
<ThreadList ThreadID="5248" StartTime="10/18/2010 6:32:21 PM" />
<ThreadList ThreadID="4940" StartTime="10/18/2010 6:32:24 PM" />
<ThreadList ThreadID="4416" StartTime="10/18/2010 6:32:27 PM" />
<ThreadList ThreadID="5928" StartTime="10/18/2010 6:32:27 PM" />
<ThreadList ThreadID="4132" StartTime="10/18/2010 6:32:27 PM" />
<ThreadList ThreadID="5944" StartTime="10/18/2010 6:32:27 PM" />
<ThreadList ThreadID="3420" StartTime="10/18/2010 6:32:30 PM" />
<ThreadList ThreadID="4404" StartTime="10/18/2010 6:32:32 PM" />
<ThreadList ThreadID="5792" StartTime="10/18/2010 6:32:34 PM" />
<ThreadList ThreadID="472" StartTime="10/19/2010 12:08:31 AM" />
<ProcessList PID="3508" ProcessName="DrgToDsc"
ProcessOwner="Livewire\Jason" StartTime="10/17/2010 10:47:57 AM"
CPUTime="00:00:00.0468003" Priority="8" StartAddress="0" />
<ThreadList ThreadID="3972" StartTime="10/19/2010 11:30:56 AM"
CPUTime="00:00:00.0468003" Priority="8" StartAddress="0" />
<ThreadList ThreadID="2208" StartTime="10/19/2010 3:19:52 PM"
CPUTime="00:00:00.0156001" Priority="10" StartAddress="2002869848" />
<ThreadList ThreadID="5948" StartTime="10/19/2010 3:53:53 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="3648" StartTime="10/19/2010 3:53:53 PM"
CPUTime="00:00:00" Priority="6" StartAddress="2002869848" />
<ThreadList ThreadID="3976" StartTime="10/19/2010 3:54:02 PM"
CPUTime="00:00:00.6864044" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="4656" StartTime="10/19/2010 3:54:03 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="1388" StartTime="10/19/2010 3:54:18 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="5456" StartTime="10/19/2010 5:25:43 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="4624" StartTime="10/19/2010 5:25:43 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="560" StartTime="10/19/2010 5:25:45 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="4776" StartTime="10/19/2010 5:25:45 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="5656" StartTime="10/19/2010 5:25:46 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="4600" StartTime="10/19/2010 5:29:46 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="520" StartTime="10/19/2010 5:29:46 PM"
CPUTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="2808" StartTime="10/19/2010 5:29:46 PM"
CPUTime="00:00:00.0312002" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="5364" StartTime="10/19/2010 5:29:46 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="4186" StartTime="10/19/2010 5:30:55 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="5372" StartTime="10/19/2010 5:31:44 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="4224" StartTime="10/19/2010 5:31:58 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="5868" StartTime="10/19/2010 5:31:58 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
</ProcessList>
<ProcessList PID="3508" ProcessName="DrgToDsc"
ProcessOwner="Livewire\Jason" StartTime="10/17/2010 10:47:57 AM"
CPUTime="00:00:00.0468003" Priority="8" StartAddress="0" />
<ThreadList ThreadID="3516" StartTime="10/17/2010 10:47:57 AM"
CPUTime="00:00:00.2652017" Priority="8" StartAddress="0" />
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
</ProcessList>
<ProcessList PID="5884" ProcessName="mstsc"
ProcessOwner="Livewire\Jason" StartTime="10/19/2010 4:25:08 PM"
CPUTime="00:00:06.9888448" PhysicalMemSize="152682496" Threads="13"
<ThreadList ThreadID="544" StartTime="10/19/2010 4:25:08 PM"
CPUTime="00:00:00.8112052" Priority="8" StartAddress="2002869848" />
CPUTime="00:00:00.8112052" Priority="8" StartAddress="2002869848" />
<table>
<thead>
<tr>
<th>ThreadID</th>
<th>Priority</th>
<th>StartAddress</th>
<th>StartTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>8</td>
<td>2002869848</td>
<td>10/17/2010 4:25:10 PM</td>
</tr>
<tr>
<td>4496</td>
<td>8</td>
<td>2002869848</td>
<td>10/17/2010 4:25:10 PM</td>
</tr>
<tr>
<td>1344</td>
<td>8</td>
<td>2002869848</td>
<td>10/17/2010 4:25:10 PM</td>
</tr>
<tr>
<td>3360</td>
<td>8</td>
<td>2002869848</td>
<td>10/17/2010 4:25:11 PM</td>
</tr>
<tr>
<td>2364</td>
<td>8</td>
<td>2002869848</td>
<td>10/17/2010 4:25:11 PM</td>
</tr>
<tr>
<td>2080</td>
<td>8</td>
<td>2002869848</td>
<td>10/17/2010 4:25:11 PM</td>
</tr>
<tr>
<td>5512</td>
<td>8</td>
<td>2002869848</td>
<td>10/17/2010 4:25:11 PM</td>
</tr>
<tr>
<td>5260</td>
<td>8</td>
<td>2002869848</td>
<td>10/17/2010 4:25:11 PM</td>
</tr>
<tr>
<td>2824</td>
<td>8</td>
<td>2002869848</td>
<td>10/17/2010 4:25:13 PM</td>
</tr>
<tr>
<td>1200</td>
<td>8</td>
<td>2002869848</td>
<td>10/17/2010 4:25:13 PM</td>
</tr>
<tr>
<td>5320</td>
<td>8</td>
<td>2002869848</td>
<td>10/17/2010 4:31:12 PM</td>
</tr>
<tr>
<td>2002869848</td>
<td>8</td>
<td>2002869848</td>
<td>10/17/2010 5:26:42 PM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PID</th>
<th>ProcessName</th>
<th>StartTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>3692</td>
<td>iTunesHelper</td>
<td>10/17/2010 10:47:54 AM</td>
</tr>
<tr>
<td>3612</td>
<td>Livewire\Jason</td>
<td>10/17/2010 10:48:10 AM</td>
</tr>
<tr>
<td>3360</td>
<td>Livewire\Jason</td>
<td>10/17/2010 10:48:10 AM</td>
</tr>
<tr>
<td>4444</td>
<td>Livewire\Jason</td>
<td>10/17/2010 10:47:54 AM</td>
</tr>
<tr>
<td>4180</td>
<td>Livewire\Jason</td>
<td>10/17/2010 11:31:00 AM</td>
</tr>
<tr>
<td>ProcessOwner</td>
<td>StartTime</td>
<td>PhysicalMemSize</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>&quot;Livewire\Jason&quot;</td>
<td>10/17/2010 10:47:48 AM</td>
<td>&quot;77230080&quot;</td>
</tr>
<tr>
<td>&quot;Livewire\Jason&quot;</td>
<td>10/17/2010 10:53:55 AM</td>
<td>&quot;350334976&quot;</td>
</tr>
</tbody>
</table>

**Thread Details**

<table>
<thead>
<tr>
<th>ThreadID</th>
<th>StartTime</th>
<th>Priority</th>
<th>StartAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;2956&quot;</td>
<td>10/19/2010 11:31:00 AM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;2296&quot;</td>
<td>10/17/2010 10:47:48 AM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;3228&quot;</td>
<td>10/17/2010 10:48:15 AM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ThreadID</th>
<th>StartTime</th>
<th>Priority</th>
<th>StartAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;3244&quot;</td>
<td>10/17/2010 10:53:55 AM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;2356&quot;</td>
<td>10/17/2010 10:53:58 AM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;3464&quot;</td>
<td>10/17/2010 10:53:58 AM</td>
<td>&quot;7&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;2128&quot;</td>
<td>10/17/2010 10:53:58 AM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;4024&quot;</td>
<td>10/17/2010 10:53:58 AM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;3628&quot;</td>
<td>10/17/2010 10:53:58 AM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;3568&quot;</td>
<td>10/17/2010 10:54:07 AM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;3736&quot;</td>
<td>10/17/2010 10:54:07 AM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;2656&quot;</td>
<td>10/17/2010 10:54:34 AM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;3088&quot;</td>
<td>10/17/2010 10:54:38 AM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;2072&quot;</td>
<td>10/17/2010 10:54:48 AM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;1260&quot;</td>
<td>10/17/2010 10:55:20 AM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;3720&quot;</td>
<td>10/17/2010 10:55:20 AM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;872&quot;</td>
<td>10/17/2010 10:57:33 AM</td>
<td>&quot;10&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;3888&quot;</td>
<td>10/19/2010 11:31:01 AM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;4712&quot;</td>
<td>10/19/2010 2:34:28 PM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;5296&quot;</td>
<td>10/19/2010 2:40:03 PM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;6116&quot;</td>
<td>10/19/2010 5:20:32 PM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>&quot;2540&quot;</td>
<td>10/19/2010 5:31:48 PM</td>
<td>&quot;8&quot;</td>
<td>&quot;2002869848&quot;</td>
</tr>
<tr>
<td>ThreadID</td>
<td>StartTime</td>
<td>Priority</td>
<td>StartAddress</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>----------</td>
<td>--------------</td>
</tr>
<tr>
<td>3392</td>
<td>10/19/2010 5:32:02 PM</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>3640</td>
<td>10/17/2010 10:47:20 AM</td>
<td>13</td>
<td>2002869848</td>
</tr>
<tr>
<td>3660</td>
<td>10/17/2010 10:47:20 AM</td>
<td>15</td>
<td>2002869848</td>
</tr>
<tr>
<td>4300</td>
<td>10/18/2010 2:59:08 PM</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>4320</td>
<td>10/17/2010 10:48:19 AM</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>1532</td>
<td>10/17/2010 10:48:19 AM</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>4236</td>
<td>10/19/2010 5:30:50 PM</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>5920</td>
<td>10/19/2010 5:31:54 PM</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>2116</td>
<td>10/17/2010 10:47:43 AM</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>2644</td>
<td>10/17/2010 10:47:45 AM</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>2780</td>
<td>10/17/2010 10:47:55 AM</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>2776</td>
<td>10/17/2010 10:47:55 AM</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>3216</td>
<td>10/17/2010 10:47:55 AM</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>3332</td>
<td>10/17/2010 10:47:55 AM</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>3360</td>
<td>10/17/2010 10:47:55 AM</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>3986</td>
<td>10/17/2010 10:48:14 AM</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>3608</td>
<td>10/17/2010 10:48:14 AM</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>5784</td>
<td>10/18/2010 3:03:36 PM</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>ProcessName</td>
<td>PID</td>
<td>PhysicalMemSize</td>
<td>Threads</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>-----------------</td>
<td>---------</td>
</tr>
<tr>
<td>igfxsrvc</td>
<td>244</td>
<td>46219264</td>
<td>20</td>
</tr>
<tr>
<td>plugin-container</td>
<td>1644</td>
<td>351674368</td>
<td>15</td>
</tr>
<tr>
<td>Livewire\Jason</td>
<td>2944</td>
<td>72249344</td>
<td>4</td>
</tr>
<tr>
<td>Livewire\Jason</td>
<td>1048</td>
<td>531674368</td>
<td>15</td>
</tr>
<tr>
<td>Livewire\Jason</td>
<td>3046</td>
<td>72249344</td>
<td>4</td>
</tr>
<tr>
<td>Livewire\Jason</td>
<td>1644</td>
<td>351674368</td>
<td>15</td>
</tr>
</tbody>
</table>

CPUTime=

00:00:00.3120020 Priority=8 StartAddress=0

00:00:00.2340015 Priority=8 StartAddress=2002869848

```xml
<ProcessList PID="2944" ProcessName="jucheck"
ProcessOwner="Livewire\Jason" StartTime="10/17/2010 10:53:11 AM"
CPUTime="00:00:00.3900025" PhysicalMemSize="72249344" Threads="4"/>

CPUTime="00:00:00.0468003" Priority="8" StartAddress="2002869848"/>

<ThreadList ThreadID="3368" StartTime="10/17/2010 10:53:36 AM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848"/>
```

```xml
<ProcessList PID="1644" ProcessName="plugin-container"
CPUTime="00:31:07.5659715" PhysicalMemSize="351674368" Threads="15"/>
```

```xml
CPUTime="00:00:33.4778146" Priority="8" StartAddress="0"/>
```

```xml
<ThreadList ThreadID="2448" StartTime="10/17/2010 10:53:12 AM"
CPUTime="00:00:00.0468003" Priority="8" StartAddress="2002869848"/>
```

```xml
<ThreadList ThreadID="3652" StartTime="10/17/2010 10:53:36 AM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848"/>
```

```xml
<ThreadList ThreadID="4280" StartTime="10/17/2010 10:28:03 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848"/>
```

```xml
<ThreadList ThreadID="4660" StartTime="10/19/2010 11:32:09 AM"
CPUTime="00:00:00.780005" Priority="8" StartAddress="2002869848"/>
```

```xml
<ThreadList ThreadID="6056" StartTime="10/19/2010 11:32:09 AM"
CPUTime="00:00:00.624004" Priority="8" StartAddress="2002869848"/>
```

```xml
<ThreadList ThreadID="4780" StartTime="10/19/2010 11:32:09 AM"
CPUTime="00:00:00.0312002" Priority="8" StartAddress="2002869848"/>
```

```xml
<ThreadList ThreadID="1672" StartTime="10/19/2010 11:32:10 AM"
CPUTime="00:00:00.0468003" Priority="8" StartAddress="2002869848"/>
```

```xml
<ThreadList ThreadID="3356" StartTime="10/19/2010 11:32:10 AM"
CPUTime="00:00:00.0156001" Priority="8" StartAddress="2002869848"/>
```

```xml
<ThreadList ThreadID="2552" StartTime="10/19/2010 11:32:10 AM"
CPUTime="00:00:00.0624004" Priority="8" StartAddress="2002869848"/>
```

```xml
<ThreadList ThreadID="5232" StartTime="10/19/2010 3:17:11 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848"/>
```

```xml
<ThreadList ThreadID="5964" StartTime="10/19/2010 3:30:25 PM"
CPUTime="00:00:00" Priority="8" StartAddress="0"/>
```

```xml
<ThreadList ThreadID="1376" StartTime="10/19/2010 3:30:50 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848"/>
```

```xml
<ThreadList ThreadID="1928" StartTime="10/19/2010 5:26:42 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848"/>
```
<ThreadList ThreadID="5888" StartTime="10/19/2010 5:27:47 PM"
CPUtilm="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="5968" StartTime="10/19/2010 5:31:49 PM"
CPUtilm="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="6128" StartTime="10/19/2010 5:32:04 PM"
CPUtilm="00:00:00" Priority="8" StartAddress="2002869848" />
</ProcessList>

<ProcessList>
  <ProcessList PID="3600" ProcessName="taskeng"
ProcessOwner="Livewire\Jason" StartTime="10/17/2010 10:47:19 AM"
CPUtilm="00:00:03.5256226" PhysicalMemSize="90001408" Threads="16">
<ThreadList ThreadID="3604" StartTime="10/17/2010 10:47:19 AM"
CPUtilm="00:00:00.0936006" Priority="8" StartAddress="0" />
<ThreadList ThreadID="3632" StartTime="10/17/2010 10:47:19 AM"
CPUtilm="00:00:00" Priority="8" StartAddress="0" />
<ThreadList ThreadID="3636" StartTime="10/17/2010 10:47:19 AM"
CPUtilm="00:00:00" Priority="8" StartAddress="2002869848"/>
<ThreadList ThreadID="3684" StartTime="10/17/2010 10:47:20 AM"
CPUtilm="00:00:00.0156001" Priority="8" StartAddress="2002869848"/>
<ThreadList ThreadID="3700" StartTime="10/17/2010 10:47:20 AM"
CPUtilm="00:00:00" Priority="8" StartAddress="2002869848"/>
CPUtilm="00:00:00" Priority="8" StartAddress="2002869848"/>
CPUtilm="00:00:00.0468003" Priority="8" StartAddress="2002869848"/>
CPUtilm="00:00:01.0140065" Priority="8" StartAddress="0" />
CPUtilm="00:00:00.8736056" Priority="8" StartAddress="2002869848"/>
CPUtilm="00:00:00.0156001" Priority="8" StartAddress="2002869848"/>
CPUtilm="00:00:00.1092007" Priority="10" StartAddress="2002869848"/>
<ThreadList ThreadID="1826" StartTime="10/17/2010 10:47:52 AM"
CPUtilm="00:00:00.4056026" Priority="8" StartAddress="2002869848"/>
<ThreadList ThreadID="1648" StartTime="10/17/2010 10:58:29 AM"
CPUtilm="00:00:00.0780005" Priority="15" StartAddress="2002869848"/>
<ThreadList ThreadID="4460" StartTime="10/19/2010 5:26:42 PM"
CPUtilm="00:00:00" Priority="8" StartAddress="2002869848"/>
<ThreadList ThreadID="3204" StartTime="10/19/2010 5:31:37 PM"
CPUtilm="00:00:00" Priority="8" StartAddress="2002869848"/>
<ThreadList ThreadID="4760" StartTime="10/19/2010 5:32:11 PM"
CPUtilm="00:00:00" Priority="8" StartAddress="2002869848"/>
</ProcessList>

<ProcessList>
  <ProcessList PID="3396" ProcessName="ApntEx"
ProcessOwner="Livewire\Jason" StartTime="10/17/2010 10:47:57 AM"
CPUtilm="00:00:03.0015859" PhysicalMemSize="55541760" Threads="3">
<ThreadList ThreadID="2932" StartTime="10/19/2010 10:47:57 AM"
CPUtilm="00:00:00.0156001" Priority="9" StartAddress="2002869848"/>
<ThreadList ThreadID="5376" StartTime="10/19/2010 10:48:04 AM"
CPUtilm="00:00:00.9859856" Priority="8" StartAddress="2002869848"/>
<ThreadList ThreadID="2064" StartTime="10/19/2010 10:48:04 AM"
CPUtilm="00:00:00" Priority="8" StartAddress="2002869848"/>
</ProcessList>

<ProcessList>
  <ProcessList PID="1620" ProcessName="Apoint"
ProcessOwner="Livewire\Jason" StartTime="10/17/2010 10:47:44 AM"
CPUtilm="00:00:05.7096366" PhysicalMemSize="66822144" Threads="2">
<ThreadList ThreadID="2508" StartTime="10/17/2010 10:47:44 AM"
<ProcessList PID="5228" ProcessName="mmc" ProcessOwner="Livewire\Jason"
StartTime="10/19/2010 4:31:58 PM" CPUMTime="00:00:00.5956551"
PhysicalMemSize="300326912" Threads="21">
  <ThreadList ThreadID="2748" StartTime="10/19/2010 4:32:01 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="4200" StartTime="10/19/2010 4:32:02 PM"
CPUMTime="00:00:00.0936006" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="1660" StartTime="10/19/2010 4:32:02 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="4472" StartTime="10/19/2010 4:31:58 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="5536" StartTime="10/19/2010 4:32:00 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="2748" StartTime="10/19/2010 4:32:01 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="4456" StartTime="10/19/2010 4:32:01 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="2748" StartTime="10/19/2010 4:32:01 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="4200" StartTime="10/19/2010 4:32:02 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="1660" StartTime="10/19/2010 4:32:02 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="4184" StartTime="10/19/2010 4:32:02 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="4368" StartTime="10/19/2010 4:32:04 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="4172" StartTime="10/19/2010 4:32:05 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="5896" StartTime="10/19/2010 4:32:05 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="5352" StartTime="10/19/2010 4:32:02 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="5448" StartTime="10/19/2010 4:32:05 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="5488" StartTime="10/19/2010 4:32:06 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="5486" StartTime="10/19/2010 4:32:06 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="4864" StartTime="10/19/2010 4:32:09 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="1892" StartTime="10/19/2010 4:44:38 PM"
CPUMTime="00:00:00.0156001" Priority="10" StartAddress="2002869848" />
  <ThreadList ThreadID="5428" StartTime="10/19/2010 4:44:38 PM"
CPUMTime="00:00:00.0156001" Priority="15" StartAddress="2002869848" />
  <ThreadList ThreadID="960" StartTime="10/19/2010 4:54:37 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="3580" StartTime="10/19/2010 5:26:42 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
</ProcessList>

<ProcessList PID="3780" ProcessName="explorer"
ProcessOwner="Livewire\Jason" StartTime="10/17/2010 10:47:23 AM"
CPUMTime="00:07:45.6629850" PhysicalMemSize="822763520" Threads="33">
CPUMTime="00:00:03.5880230" Priority="8" StartAddress="2002869848" />
  <ThreadList ThreadID="4048" StartTime="10/17/2010 10:47:34 AM"
CPUMTime="00:02:43.0210450" Priority="9" StartAddress="0" />
</ProcessList>
<ThreadList ThreadID="4056" StartTime="10/17/2010 10:47:34 AM"
CPUTime="00:00:10.0776646" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="2196" StartTime="10/17/2010 10:47:36 AM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="2088" StartTime="10/17/2010 10:47:50 AM"
CPUTime="00:00:00.0624004" Priority="15" StartAddress="2002869848" />
<ThreadList ThreadID="768" StartTime="10/17/2010 10:47:50 AM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="2220" StartTime="10/17/2010 10:47:52 AM"
CPUTime="00:00:00.073632472" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="2800" StartTime="10/17/2010 10:47:54 AM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="2804" StartTime="10/17/2010 10:47:54 AM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="1856" StartTime="10/17/2010 10:47:54 AM"
CPUTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="3672" StartTime="10/17/2010 10:48:00 AM"
CPUTime="00:00:00.0468003" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="2968" StartTime="10/17/2010 10:48:00 AM"
CPUTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
CPUTime="00:00:00.0468003" Priority="8" StartAddress="2002869848" />
CPUTime="00:00:22.9009466" Priority="15" StartAddress="2002869848" />
CPUTime="00:00:00.0624004" Priority="8" StartAddress="0" />
CPUTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="3656" StartTime="10/17/2010 10:48:19 AM"
CPUTime="00:00:00.0468003" Priority="8" StartAddress="2002869848" />
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="2112" StartTime="10/17/2010 10:48:40 AM"
CPUTime="00:00:00.0156001" Priority="8" StartAddress="2002869848" />
CPUTime="00:00:00.7332047" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="1032" StartTime="10/17/2010 10:58:47 AM"
CPUTime="00:00:00.0780005" Priority="10" StartAddress="2002869848" />
<ThreadList ThreadID="3920" StartTime="10/17/2010 5:55:35 PM"
CPUTime="00:00:00.03120002" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="2188" StartTime="10/19/2010 11:32:09 AM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="4304" StartTime="10/19/2010 4:36:17 PM"
CPUTime="00:00:02.9016186" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="6048" StartTime="10/19/2010 4:36:18 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="4724" StartTime="10/19/2010 4:36:18 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="5292" StartTime="10/19/2010 4:36:18 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="3440" StartTime="10/19/2010 4:36:18 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="660" StartTime="10/19/2010 5:26:42 PM"
CPUTime="00:00:00" Priority="8" StartAddress="2002869848" />
<ThreadList ThreadID="3176" StartTime="10/19/2010 5:26:45 PM"
<table>
<thead>
<tr>
<th>ProcessName</th>
<th>PhysicalMemSize</th>
<th>Priority</th>
<th>StartAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Livewire\Jason</td>
<td>3927654</td>
<td>8</td>
<td>200433664</td>
</tr>
<tr>
<td>SIMS_Client.vshost</td>
<td>2002869848</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>scheduler_proxy</td>
<td>2002869848</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>Livewire\Jason</td>
<td>66240512</td>
<td>8</td>
<td>2002869848</td>
</tr>
<tr>
<td>GrooveMonitor</td>
<td>66596864</td>
<td>8</td>
<td>2002869848</td>
</tr>
</tbody>
</table>
<ProcessList PID="1796" ProcessName="WINWORD"

ProcessOwner="Livewire\Jason" StartTime="10/17/2010 10:59:49 AM"
CPUMTime="01:27:23.8800144" PhysicalMemSize="72930992" Threads="7">
  <ThreadList ThreadID="3696" StartTime="10/17/2010 10:59:49 AM"
CPUMTime="01:25:47.7209980" Priority="8" StartAddress="0"/>
  <ThreadList ThreadID="1864" StartTime="10/17/2010 11:00:30 AM"
CPUMTime="00:00:00" Priority="8" StartAddress="2002869846"/>
  <ThreadList ThreadID="3400" StartTime="10/17/2010 11:00:37 AM"
CPUMTime="00:00:00" Priority="8" StartAddress="2002869846"/>
  <ThreadList ThreadID="532" StartTime="10/17/2010 11:17:59 AM"
CPUMTime="00:00:00.5148033" Priority="8" StartAddress="2002869848"/>
  <ThreadList ThreadID="5732" StartTime="10/19/2010 5:18:20 PM"
CPUMTime="00:00:00.1248008" Priority="8" StartAddress="2002869848"/>
  <ThreadList ThreadID="4240" StartTime="10/19/2010 5:22:46 PM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848"/>
  <ThreadList ThreadID="4128" StartTime="10/19/2010 5:30:19 PM"
CPUMTime="00:00:00" Priority="8" StartAddress="2002869848"/>
</ProcessList>

</ProcessList>

<ProcessList PID="2372" ProcessName="PMHandler"

ProcessOwner="Livewire\Jason" StartTime="10/17/2010 10:47:43 AM"
CPUMTime="00:00:04.5552292" PhysicalMemSize="68284416" Threads="5">
  <ThreadList ThreadID="1340" StartTime="10/17/2010 10:47:43 AM"
CPUMTime="00:00:04.5396291" Priority="8" StartAddress="2002869848"/>
  <ThreadList ThreadID="2848" StartTime="10/17/2010 10:47:54 AM"
CPUMTime="00:00:00.0156001" Priority="8" StartAddress="2002869848"/>
  <ThreadList ThreadID="2840" StartTime="10/17/2010 10:47:54 AM"
CPUMTime="00:00:00" Priority="8" StartAddress="0"/>
  <ThreadList ThreadID="2924" StartTime="10/17/2010 10:47:54 AM"
CPUMTime="00:00:00" Priority="8" StartAddress="0"/>
  <ThreadList ThreadID="2832" StartTime="10/17/2010 10:47:54 AM"
CPUMTime="00:00:00" Priority="8" StartAddress="2002869848"/>
</ProcessList>

</ProcessList>

<ProcessList PID="4928" ProcessName="cmd" ProcessOwner="Livewire\Jason"

StartTime="10/17/2010 10:59:49 AM" CPUMTime="00:00:00.2652017"
PhysicalMemSize="23166976" Threads="1">
  <ThreadList ThreadID="5376" StartTime="10/17/2010 10:59:49 AM"
CPUMTime="00:00:00.2652017" Priority="8" StartAddress="2002869848"/>
</ProcessList>

</Snapshots>
<xsd:/schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
  <xsd:annotation>
    <xsd:appinfo>
      <sql:relationship name="SnapshotsToDirectories">
        <sql:parent>"Snapshots"</sql:parent>
        <sql:parent-key>"SnapID"</sql:parent-key>
        <sql:child>"Directories"</sql:child>
        <sql:child-key>"DirectoryID"</sql:child-key>
      </sql:relationship>
      <sql:relationship name="DirectoriesToFiles">
        <sql:parent>"Directories"</sql:parent>
        <sql:parent-key>"DirectoryID"</sql:parent-key>
        <sql:child>"Files"</sql:child>
        <sql:child-key>"DirectoryID"</sql:child-key>
      </sql:relationship>
      <sql:relationship name="FilesToADSData">
        <sql:parent>"Files"</sql:parent>
        <sql:parent-key>"FileID"</sql:parent-key>
        <sql:child>"AdsData"</sql:child>
        <sql:child-key>"FileID"</sql:child-key>
      </sql:relationship>
      <sql:relationship name="SnapshotsToActTcpConnections">
        <sql:parent>"Snapshots"</sql:parent>
        <sql:parent-key>"SnapID"</sql:parent-key>
        <sql:child>"ActiveTcpConnections"</sql:child>
        <sql:child-key>"ProfileID"</sql:child-key>
      </sql:relationship>
      <sql:relationship name="SnapshotsToTcpListConnections">
        <sql:parent>"Snapshots"</sql:parent>
        <sql:parent-key>"SnapID"</sql:parent-key>
        <sql:child>"TcpListeningConnections"</sql:child>
        <sql:child-key>"ProfileID"</sql:child-key>
      </sql:relationship>
      <sql:relationship name="SnapshotsToUdpListConnections">
        <sql:parent>"Snapshots"</sql:parent>
        <sql:parent-key>"SnapID"</sql:parent-key>
        <sql:child>"UdpListeningConnections"</sql:child>
        <sql:child-key>"ProfileID"</sql:child-key>
      </sql:relationship>
      <sql:relationship name="SnapshotsToProcessList">
        <sql:parent>"Snapshots"</sql:parent>
        <sql:parent-key>"SnapID"</sql:parent-key>
        <sql:child>"ProcessList"</sql:child>
        <sql:child-key>"ProfileID"</sql:child-key>
      </sql:relationship>
      <sql:relationship name="ProcessListToThreadList">
        <sql:parent>"ProcessList"</sql:parent>
        <sql:parent-key>"ProcID"</sql:parent-key>
        <sql:child>"ThreadList"</sql:child>
        <sql:child-key>"ProcID"</sql:child-key>
      </sql:relationship>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:element name="Snapshots" sql:relation="Snapshots" sql:key-fields="SnapID">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Directories" sql:relation="Directories" sql:key-fields="SnapID DirectoryID" sql:relationship="SnapshotsToDirectories">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Files" sql:relation="Files" sql:key-fields="DirectoryID FileID" sql:relationship="DirectoriesToFiles">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="AdsData" sql:relation="AdsData" sql:key-fields="FileID" sql:relationship="FilesToADSData">
                      <!-- Additional elements and relationships -->
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
<xsd:complexType>
  <xsd:attribute name="FileID" type="xsd:int"/>
  <xsd:attribute name="ADSPath" type="xsd:string"/>
  <xsd:attribute name="ADSSize" type="xsd:int"/>
</xsd:complexType>

<xsd:complexType>
  <xsd:attribute name="DirectoryID" type="xsd:int"/>
  <xsd:attribute name="FileID" type="xsd:int"/>
  <xsd:attribute name="FileName" type="xsd:string"/>
  <xsd:attribute name="LastWriteTime" type="xsd:string"/>
  <xsd:attribute name="LastAccessTime" type="xsd:string"/>
  <xsd:attribute name="FileCreationTime" type="xsd:string"/>
  <xsd:attribute name="FileSize" type="xsd:int"/>
  <xsd:attribute name="FileMD5" type="xsd:string"/>
  <xsd:attribute name="FileADS" type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType>
  <xsd:attribute name="DirectoryID" type="xsd:int"/>
  <xsd:attribute name="SnapID" type="xsd:int"/>
  <xsd:attribute name="DirectoryName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType>
  <xsd:attribute name="SnapID" type="xsd:int"/>
  <xsd:attribute name="LocalTCPConnection" type="xsd:string"/>
  <xsd:attribute name="RemoteTCPConnection" type="xsd:string"/>
  <xsd:attribute name="TcpConnectionState" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType>
  <xsd:attribute name="SnapID" type="xsd:int"/>
  <xsd:attribute name="LocalTCPConnection" type="xsd:string"/>
  <xsd:attribute name="RemoteTCPConnection" type="xsd:string"/>
  <xsd:attribute name="TcpConnectionState" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType>
  <xsd:attribute name="SnapID" type="xsd:int"/>
</xsd:complexType>

<xsd:complexType>
  <xsd:attribute name="SnapID" type="xsd:int"/>
  <xsd:attribute name="TcpConnectionState" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType>
  <xsd:attribute name="SnapID" type="xsd:int"/>
</xsd:complexType>

<xsd:complexType>
  <xsd:attribute name="TcpConnectionState" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType>
  <xsd:attribute name="SnapID" type="xsd:int"/>
</xsd:complexType>

<xsd:complexType>
  <xsd:attribute name="TcpConnectionState" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType>
  <xsd:attribute name="SnapID" type="xsd:int"/>
</xsd:complexType>

<xsd:complexType>
  <xsd:attribute name="TcpConnectionState" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType>
  <xsd:attribute name="SnapID" type="xsd:int"/>
</xsd:complexType>

<xsd:complexType>
  <xsd:attribute name="TcpConnectionState" type="xsd:string"/>
</xsd:complexType>
sql:relationship="SnapshotsToTcpListConnections">
  <xsd:complexType>
    <xsd:attribute name="SnapID" type="xsd:int"/>
    <xsd:attribute name="TCPEntry" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
sql:relation="UdpListeningConnections" sql:key-fields="SnapID"
sql:relationship="SnapshotsToUdpListConnections">
  <xsd:complexType>
    <xsd:attribute name="SnapID" type="xsd:int"/>
    <xsd:attribute name="UDPEntry" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
sql:relation="ProcessList" sql:key-fields="SnapID ProcID"
sql:relationship="SnapshotsToProcessList">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="ThreadList" sql:relationship="ProcessListToThreadList">
        <xsd:complexType>
          <xsd:attribute name="ProcID" type="xsd:int"/>
          <xsd:attribute name="ThreadID" type="xsd:int"/>
          <xsd:attribute name="StartTime" type="xsd:string"/>
          <xsd:attribute name="CPUTime" type="xsd:string"/>
          <xsd:attribute name="Priority" type="xsd:int"/>
          <xsd:attribute name="StartAddress" type="xsd:string"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="SnapID" type="xsd:int"/>
    <xsd:attribute name="ProcID" type="xsd:int"/>
    <xsd:attribute name="PID" type="xsd:int"/>
    <xsd:attribute name="ProcessName" type="xsd:string"/>
    <xsd:attribute name="ProcessOwner" type="xsd:string"/>
    <xsd:attribute name="StartTime" type="xsd:string"/>
    <xsd:attribute name="CPUTime" type="xsd:string"/>
    <xsd:attribute name="PhysicalMemSize" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
</xsd:relationship>
<xsd:complexType name="log">
  <xsd:complexContent>
    <xsd:restriction base="xsd:sequence">
      <xsd:sequence>
        <xsd:element name="SnapID" type="xsd:int"/>
        <xsd:element name="SnapDate" type="xsd:string"/>
        <xsd:element name="ClientMAC" type="xsd:string"/>
        <xsd:element name="IPAddress" type="xsd:string"/>
        <xsd:element name="Hostname" type="xsd:string"/>
        <xsd:element name="Threads" type="xsd:int"/>
      </xsd:sequence>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
/*
 * Name: Jason Batchelor
 * Program: CAIT
 * Date: 10/3/2010
 *
 * This program is a proof of concept being done for the partial fulfillment
 * of the MS in Networking and Systems
 * Administration from the Rochester Institute of Technology. All rights
 * reserved.
 *
 * Builder.cs: Builds the snapshot XML file for the client by calling child
 * classes
 * and gathering aggregate information.
 */

using System;
using System.IO;
using System.Text;
using System.Net;
using System.Net.NetworkInformation;
using System.Xml;
using System.Management;
using System.Xml.Serialization;
using System.Diagnostics;
using System.Linq;

public class Builder
{
    public static void buildXML(String[] dirs)
    {
        // Initiate network collection objects
        IPGlobalProperties ipProperties = IPGlobalProperties.GetIPGlobalProperties();
        ManagementClass mc = new ManagementClass("Win32_NetworkAdapterConfiguration");
        ManagementObjectCollection moc = mc.GetInstances();
        // Use IP and Management objects to create NetworkAggregation object
        NetworkAggregation netfo = new NetworkAggregation(ipProperties, moc);

        // Begin the creation of the system snapshot file, called sys.xml
        XmlTextWriter writer = new XmlTextWriter("sys.xml", null);
        writer.Formatting = Formatting.Indented;
        writer.WriteStartDocument();

        // Direct aggregation types are broken up into respective functions
        // enumClient gathers client attributes for the snapshot database
        enumClient(writer, netfo);
        // enumDirs gathers directory information and calls respective
        enumFiles function
        enumDirs(writer, dirs);
// enumActiveTcp gather information about active tcp connections
enumActiveTcp(writer, netfo);
// enumListenTcp gather information about listening tcp connections
enumListenTcp(writer, netfo);
// enumListenUdp gather information about listening udp connections
enumListenUdp(writer, netfo);
// enumProcs gather information on current processes and calls a
function to enumerate process threads
enumProcs(writer);

// Finally, end the last xml element and close the file gracefully
writer.WriteEndElement();
writer.WriteEndDocument();
writer.Flush();
writer.Close();

public static void enumClient(XmlTextWriter writer, NetworkAggregation netfo)
{
    // Gather data for the snapshots table
    writer.WriteStartElement("Snapshots");
    writer.WriteAttributeString("SnapDate", DateTime.Now.ToString());
    writer.WriteAttributeString("ClientMAC", netfo.macAddress);
    writer.WriteAttributeString("IPAddress", netfo.ipAddress);
    writer.WriteAttributeString("Hostname", netfo.hostName);
}

public static void enumDirs(XmlTextWriter writer, String[] dirs)
{
    // Begin recursively searching through each directory in the array
    for (int i = 0; i < dirs.Length; i++)
    {
        if (Directory.Exists(dirs[i])) // Evaluate the directory to
        ensure it exists
        {
            enumFiles(writer, dirs[i]); // Enumerate all file metadata in
            each corresponding directory
            foreach (String dirName in Directory.GetDirectories(dirs[i],
"*", SearchOption.AllDirectories))
            {
                enumFiles(writer, dirName);
            }
        }
    }
}

public static void enumFiles(XmlTextWriter writer, String dirName)
{
    // Begin enumerating the file metadata for each directory
    DirectoryInfo dir = new DirectoryInfo(dirName);
    // Write directory attributes
    writer.WriteStartElement("Directories");
    writer.WriteAttributeString("DirectoryName", dirName);
    try
    {

dir.GetFiles("*");
}
catch (DirectoryNotFoundException e)
{
    EventLog.WriteEntry("CaitO Service", e.Message, EventLogEntryType.Warning);
}

FileInfo[] fileList = dir.GetFiles("*");
foreach (FileInfo file in fileList)
{
    FileAggregation meta = new FileAggregation(file);
    // Gather file data and write to respective xml attribute
    writer.WriteStartElement("Files");
    writer.WriteAttributeString("FileName", meta.fileName);
    writer.WriteAttributeString("LastWriteTime", meta.lastWriteTime.ToString());
    writer.WriteAttributeString("LastAccessTime", meta.lastAccessTime.ToString());
    writer.WriteAttributeString("FileCreationTime", meta.fileCreationTime.ToString());
    writer.WriteAttributeString("FileMD5", meta.fileMd5);
    if (meta.fileADS.Count > 0)
    {
        // If alternate data streams are detected, they need to be processed
        writer.WriteAttributeString("FileADS", "1");
        enumADS(writer, meta);
    }
    else
    {
        writer.WriteAttributeString("FileADS", "0");
    }
    writer.WriteEndElement(); // End file element
}

public static void enumADS(XmlTextWriter writer, FileAggregation meta)
{
    // Process alternate data streams
    String[] fileAds = new String[] { meta.fileADS.ToArray(typeof(String)));
    for (int x = 0; x < fileAds.Length; x++)
    {
        writer.WriteStartElement("AdsData");
        writer.WriteAttributeString("ADSPath", fileAds[x].Substring(0, fileAds[x].IndexOf(""))).
        writer.WriteAttributeString("ADSSize", fileAds[x].Substring(fileAds[x].IndexOf("")) + 1));
        writer.WriteEndElement(); // End ads element
public static void enumActiveTcp(XmlTextWriter writer, NetworkAggregation netfo) {
    // Gather information on active tcp connections and write to xml file
    String[] activeTcpConnections = netfo.activeTcpConnections;
    for (int i = 0; i < activeTcpConnections.Length; i++) {
        writer.WriteStartElement("ActiveTcpConnections");
        writer.WriteAttributeString("LocalTCPConnection", activeTcpConnections[i].Substring(0, activeTcpConnections[i].IndexOf('*')));
        writer.WriteAttributeString("RemoteTCPConnection", activeTcpConnections[i].Substring(activeTcpConnections[i].IndexOf('*') + 1, activeTcpConnections[i].IndexOf('*') - 1 - activeTcpConnections[i].IndexOf('"')));
        writer.WriteAttributeString("TcpConnectionState", activeTcpConnections[i].Substring(activeTcpConnections[i].LastIndexOf('"') + 1));
        writer.WriteEndElement(); // End active tcp element
    }
}

public static void enumListenTcp(XmlTextWriter writer, NetworkAggregation netfo) {
    // Gather information on listening tcp connections and write to xml file
    String[] tcpListening = netfo.tcpListening;
    for (int i = 0; i < tcpListening.Length; i++) {
        writer.WriteStartElement("TcpListeningConnections");
        writer.WriteAttributeString("TCPEntry", tcpListening[i]);
        writer.WriteEndElement(); // End listening tcp element
    }
}

public static void enumListenUdp(XmlTextWriter writer, NetworkAggregation netfo) {
    // Gather information on listening udp connections and write to xml file
    String[] udpListening = netfo.udpListening;
    for (int i = 0; i < udpListening.Length; i++) {
        writer.WriteStartElement("UdpListeningConnections");
        writer.WriteAttributeString("UDPEntry", udpListening[i]);
        writer.WriteEndElement(); // End listening udp element
    }
}

public static void enumProcs(XmlTextWriter writer) {
    // Gather information on running processes and write to xml file
    foreach (Process proc in Process.GetProcesses().Where(
    )}
proc =>
{
    bool hasException = false;
    try { IntPtr x = proc.Handle; }
    catch { hasException = true; }
    return !hasException;
}) // Check for exceptions handling processes list
{
    ProcessAggregation procfo = new ProcessAggregation(proc);
    writer.WriteStartElement("ProcessList");
    writer.WriteAttributeString("PID", procfo.procID.ToString());
    writer.WriteAttributeString("ProcessName", procfo.procName);
    writer.WriteAttributeString("ProcessOwner", procfo.procOwner);
    writer.WriteAttributeString("StartTime", procfo.startTime.ToString());
    writer.WriteAttributeString("CPUTime", procfo.cpuTime.ToString());
    writer.WriteAttributeString("Threads", procfo.threadCount.ToString());
    // Call function to aggregate info on each thread in the process
    enumThreads(writer, proc);
    writer.WriteEndElement(); // End process element
}

public static void enumThreads(XmlTextWriter writer, Process proc)
{
    // Gather information on each thread
    ProcessThreadCollection threads = proc.Threads;
    foreach (ProcessThread pt in threads)
    {
        ThreadAggregation threadfo = new ThreadAggregation(pt);
        writer.WriteStartElement("ThreadList");
        writer.WriteAttributeString("ThreadID", threadfo.threadID.ToString());
        writer.WriteAttributeString("StartTime", threadfo.startTime.ToString());
        writer.WriteAttributeString("CPUTime", threadfo.cpuTime.ToString());
        writer.WriteAttributeString("Priority", threadfo.priority.ToString());
        writer.WriteAttributeString("StartAddress", threadfo.startAddress.ToString());
        writer.WriteEndElement(); // End thread element
    }
}
/*
* Name: Jason Batchelor
* Program: CAIT
* Date: 10/3/2010
*
* This program is a proof of concept being done for the partial fulfillment
of the MS in Networking and Systems
* Administration from the Rochester Institute of Technology. All rights
reserved.
*
* CaitO: This class provides the template for invoking the agent as a
service in Windows.
*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.ServiceProcess;
using System.Text;
using System.Threading;

namespace CaitO
{
    class CaitO : ServiceBase
    {

        // The main entry point for the application.
        static void Main()
        {
            // Main runtime for CaitO program
            ServiceBase.Run(new CaitO());
        }

        public CaitO()
        {
            // Specify service name
            this.ServiceName = "CaitO";
        }

        protected override void OnStart(String[] args)
        {
            // Create separate thread to speed up service and run
            ThreadStart job = new ThreadStart(Client.onStart);
            Thread thread = new Thread(job);
            thread.Start();
        }

        protected override void OnStop()
        {
            // Stop the service
            base.OnStop();
        }
    }
}
using System;
using System.IO;
using Microsoft.Win32;
using System.Collections;
using System.Threading;

public class Client
{
    public static void onStart()
    {
        // Gather runtime information from the registry
        RegistryKey regkey = Registry.LocalMachine.CreateSubKey(@"SOFTWARE\SIMSClient");
        RegQuery query = new RegQuery(regkey);

        // Access instruction file and dump contents into array list
        ArrayList targets = DataConnection.getTargets(query.serverName, query.refFileName);
        regkey.Close();

        String runTime = targets[0].ToString();
        targets.RemoveAt(0);
        // Determine if program can be run
        myTime(runTime, targets, query);

        onStart();
    }

    public static void myTime(String runTime, ArrayList targets, RegQuery query)
    {
        String currentTime = DateTime.Now.TimeOfDay.ToString().Substring(0, 5);

        if (currentTime.Equals(runTime)) // If it is time for agent to report, invoke necessary functions
        {
            Object dbServer = targets[0];
            targets.RemoveAt(0);
        }
    }
}
Object dbName = targets[0];
targets.RemoveAt(0);

Random rand = new Random();
Thread.Sleep(rand.Next(30000, 60000)); // Random backoff from thirty seconds to a minute

String[] dirs = (String[])targets.ToArray(typeof(String)); // Build directory array

Builder.buildXML(dirs); // Generate system snapshot file

DataConnection.retriever(query.serverName); // Pull down schema file

DataConnection.createDBThread(dbServer, dbName); // Upload snapshot file with schema definition

} else {
    Thread.Sleep(60000); // Backoff for a minute if run time is incorrect
    onStart();

}
using System;
using System.Data.SqlClient;
using System.Net;
using System.IO;
using System.Threading;
using System.Xml;
using System.Collections;
using System.Diagnostics;

public class DataConnection
{
    public static void retriever(String serverName)
    {
        // Grab the schema definition file
        WebClient client = new WebClient();
        try
        {
            client.Dispose();
        }
        catch (Exception e)
        {
            EventLog.WriteEntry("CaitO Service", e.Message, EventLogEntryType.Warning);
            retriever(serverName); // If there is an error, retry
        }
    }

    public static ArrayList getTargets(String serverName, String fileName)
    {
        // Get instruction list and process each element
        ArrayList targetArr = new ArrayList();
        String URLPath = "http://" + serverName + "/" + fileName;
        XmlTextReader myXmlURLreader = new XmlTextReader(URLPath);
        while (myXmlURLreader.Read())
        {
            if (!myXmlURLreader.ReadString().Equals("\r\n"))
            {
public static void createDBThread(Object dbServer, Object dbName)
{
    // Create a thread and invoke the bulk upload process
    Thread bulkLoad = new Thread(delegate()
    { uploadBulkXML(dbServer, dbName); });
    bulkLoad.SetApartmentState(ApartmentState.STA);
    bulkLoad.Start();
}

private static void uploadBulkXML(Object dbServer, Object dbName)
{
    // Connect to the database and upload the system snapshot file with the appropriate schema definition
    String serverUID = "SIMSCli";
    String serverPwd = "vmware";

    try {
        // create the bulk load object to load the schema and data files to the db
        SQLXMLBULKLOADLib.SQLXMLBulkLoad4Class objBL = new SQLXMLBULKLOADLib.SQLXMLBulkLoad4Class();
        /* To implement when integrated with a domain.
        objBL.ConnectionString = "Provider=sqloledb;server=" + server + 
        ";" +
        "database=" + dbName + ";" +
        "integrated security=SSPI";
        */
        objBL.ConnectionString = "Provider=sqloledb;server=" + dbServer + 
        ";" +
        "database=" + dbName + ";" +
        "uid=" + serverUID + ";" +
        "pwd=" + serverPwd;

        objBL.ErrorLogFile = "error.xml";
        objBL.KeepIdentity = false;
        objBL.Execute("UniversalSchema.xsd", "sys.xml"); // Bulk database upload
    }
    catch (Exception e)
    {
        EventLog.WriteEntry("CaitO Service", e.Message, EventLogEntryType.Warning);
    }
    cleanup(); // Erase files from the system
public static void cleanup()
{
    if (File.Exists("sys.xml") && File.Exists("UniversalSchema.xsd"))
    {
        File.Delete("sys.xml");
        File.Delete("UniversalSchema.xsd");
    }
}
using System;
using System.IO;
using System.Text;
using Trinet.Core.IO.Ntfs;
using System.Collections;
using System.Diagnostics;

public class FileAggregation
{
    public String fileName;
    public DateTime lastWriteTime;
    public DateTime lastAccessTime;
    public DateTime fileCreationTime;
    public long fileSize;
    public String fileMd5;
    public ArrayList fileADS;

    public FileAggregation(FileInfo file)
    {
        fileName = getName(file);
        lastWriteTime = getLastWriteTime(file);
        lastAccessTime = getLastAccessTime(file);
        fileCreationTime = getCreationTime(file);
        fileSize = getSize(file);
        fileMd5 = getMD5(file.FullName);
        fileADS = altDataStream(file);
    }

    public static String getName(FileInfo file)
    {
        return file.Name;
    }

    public static DateTime getLastWriteTime(FileInfo file)
    {
        return file.LastWriteTimeUtc;
    }

    public static DateTime getLastAccessTime(FileInfo file)
    {
        return file.LastAccessTimeUtc;
    }

    public static DateTime getCreationTime(FileInfo file)
    {
        return file.CreationTime;
    }

    public static long getSize(FileInfo file)
    {
        return file.Length;
    }

    public static String getMD5(String file)
    {
        return fileMd5;
    }

    public static ArrayList altDataStream(FileInfo file)
    {
        return fileADS;
    }
}
```csharp
public static DateTime getCreationTime(FileInfo file)
{
    return file.CreationTimeUtc;
}

public static long getFanSize(FileInfo file)
{
    return file.Length;
}

public static String getMD5(String fileName)
{
    // Examine file and enumerate hash
    try
    {
        StringBuilder sb = new StringBuilder();
        FileStream fs = new FileStream(fileName, FileMode.Open, FileAccess.Read);
        MD5 md5 = new MD5CryptoServiceProvider();
        byte[] hash = md5.ComputeHash(fs);
        fs.Close();
        foreach (byte hex in hash)
        {
            sb.Append(hex.ToString("x2"));
        }
        return sb.ToString();
    }
    catch (UnauthorizedAccessException e)
    {
        EventLog.WriteEntry("CaitO Service", e.Message, EventLogEntryType.Warning);
        return "Unauthorized: Must have read permission to get MD5."
    }
    catch (IOException e)
    {
        EventLog.WriteEntry("CaitO Service", e.Message, EventLogEntryType.Warning);
        return "IOException: File in use by another process when scanned.";
    }
}

public static ArrayList altDataStream(FileInfo file)
{
    // Invoke third party dll to retrieve ads information
    ArrayList altDataStream = new ArrayList();
    foreach (AlternateDataStreamInfo s in file.ListAlternateDataStreams())
    {
        altDataStream.Add(file.FullName + ":" + s.Name + ":" + s.Size);
    }
    return altDataStream;
}
```
using System;
using System.IO;
using System.Text;
using System.Net;
using System.Net.NetworkInformation;
using System.Management;

public class NetworkAggregation
{
    public String hostName;
    public String ipAddress;
    public String macAddress;
    public String[] activeTcpConnections;
    public String[] tcpListening;
    public String[] udpListening;

    public NetworkAggregation(IPGlobalProperties ipProperties, ManagementObjectCollection moc)
    {
        hostName = getHostName(ipProperties);
        ipAddress = getIpAddress(moc);
        macAddress = getMacAddress(moc);
        activeTcpConnections = getActiveTcpConnections(ipProperties);
        tcpListening = getTcpListening(ipProperties);
        udpListening = getUdpListening(ipProperties);
    }

    public static String getHostName(IPGlobalProperties ipProperties)
    {
        return ipProperties.HostName;
    }

    public static String getMacAddress(ManagementObjectCollection moc)
    {
        String activeMacAddress = null;
        foreach (ManagementObject mo in moc)
        {
            if (activeMacAddress == null)
            { // Get mac address of the first ip enabled nic
                if ((bool)mo["IPEnabled"] == true) activeMacAddress = mo["MacAddress"].ToString();
            }
        }
        return activeMacAddress;
    }
}

/*
 * Name: Jason Batchelor
 * Program: C
 * Date: 10/3/2010
 * This program is a proof of concept being done for the partial fulfillment of the MS in Networking and Systems
 * Administration from the Rochester Institute of Technology. All rights reserved.
 * NetworkAggregation.cs: Handles all accessor methods for client socket and connection information.
 */
public static string getIpAddress(ManagementObjectCollection moc)
{
    string[] ipAddressList = null;
    foreach (ManagementObject mo in moc)
    {
        if (ipAddressList == null)
        {
            // Get the ip address of the first ip enabled nic
            if ((bool)mo["IPEnabled"] == true) ipAddressList = (string[])mo["IPAddress"];
        }
        mo.Dispose();
    }
    return ipAddressList[0];
}

public static string[] getActiveTcpConnections(IPGlobalProperties ipProperties)
{
    // Return an array of all the active tcp connections present on the system
    int i = 0;
    tcpConnectionInformation[] tcpConnections = ipProperties.getActiveTcpConnections();
    string[] activeTcpInfo = new string[tcpConnections.Length];
    foreach (TcpConnectionInformation info in tcpConnections)
    {
        i++;
    }
    return activeTcpInfo;
}

public static string[] getTcpListening(IPGlobalProperties ipProperties)
{
    // Return a string array of listening tcp connections
    int i = 0;
    IPEndPoint[] tcpListeners = ipProperties.getActiveTcpListeners();
    string[] listeningTcpInfo = new string[tcpListeners.Length];
    foreach (IPEndPoint info in tcpListeners)
    {
        listeningTcpInfo[i] = (info.Address.ToString() + ":" + info.Port.ToString());
        i++;
    }
    return listeningTcpInfo;
}
public static String[] getUdpListening(IPGlobalProperties ipProperties) {
    // Return a string array of listening udp connections
    int i = 0;

    IPEndPoint[] udpListeners = ipProperties.GetActiveUdpListeners();
    String[] listeningUdpInfo = new String[udpListeners.Length];
    foreach (IPEndPoint info in udpListeners) {
        listeningUdpInfo[i] =
            (info.Address.ToString() + ":" + info.Port.ToString());
        i++;
    }
    return listeningUdpInfo;
}
using System;
using System.Diagnostics;
using System.Management;

public class ProcessAggregation
{
    public int procID;
    public String procName;
    public String procOwner;
    public DateTime startTime;
    public TimeSpan cpuTime;
    public long physMemSize;
    public int threadCount;

    public ProcessAggregation(Process proc)
    {
        procID = getID(proc);
        procName = getName(proc);
        procOwner = getProcessOwner(proc);
        startTime = getStartTime(proc);
        cpuTime = getCpuTime(proc);
        physMemSize = getPhysMemAlloc(proc);
        threadCount = getThreadCount(proc);
    }

    public static String getName(Process proc)
    {
        return proc.ProcessName;
    }

    public static int getID(Process proc)
    {
        return proc.Id;
    }

    public static DateTime getStartTime(Process proc)
    {
        return proc.StartTime;
    }

    public static TimeSpan getCpuTime(Process proc)
    {
        return proc.CpuTime;
    }

    public static TimeSpan getPhysMemAlloc(Process proc)
    {
        return proc.PhysicalMemAlloc;
    }
{   return proc.TotalProcessorTime;
}

public static long getPhysicaMemAlloc(Process proc)
{
   return proc.VirtualMemorySize64;
}

public static int getThreadCount(Process proc)
{
   return proc.Threads.Count;
}

/* Code provided by www.stackoverflow.com
 * Full URL: http://stackoverflow.com/questions/777548/how-do-i-
determine-the-owner-of-a-process-in-c
 * There is no method in .NET that will allow you to determine the owner
of a process. Therefore,
 * one needs to make a craft a WMI query to retrieve that information.
 */

public string getProcessOwner(Process proc)
{
   // Query process and get all relevant information, then pull the
process owner from the result.
   string query = "Select * From Win32_Process Where ProcessID = " +
proc.Id;
   ManagementObjectSearcher searcher = new
   ManagementObjectSearcher(query);
   ManagementObjectCollection processList = searcher.Get();
   foreach (ManagementObject obj in processList)
   {
      string[] argList = new string[] { string.Empty, string.Empty };
      int returnVal = Convert.ToInt32(obj.InvokeMethod("GetOwner",
argList));
      if (returnVal == 0)
      {
         return argList[1] + "\" + argList[0];
      }
   }
   return "NO OWNER";
}
RegQuery.cs

/*
* Name: Jason Batchelor
* Program: CAIT
* Date: 10/3/2010
*
* This program is a proof of concept being done for the partial fulfillment
of the MS in Networking and Systems
* Administration from the Rochester Institute of Technology. All rights
reserved.
* RegQuery.cs: Class handles the retrieval of elements from the registry
required for the program to run.
*/

using System;
using Microsoft.Win32;
using System.Diagnostics;

public class RegQuery
{
    public String serverName;
    public String refFileName;

    public RegQuery(RegistryKey regkey)
    {
        serverName = getServerName(regkey);
        refFileName = getRefFileName(regkey);
    }

    public static String getServerName(RegistryKey regkey)
    {
        // Get the server location ip address or domain
        String myServerName = null;
        try
        {
            myServerName = regkey.GetValue("ServerLoc").ToString();
        }
        catch (Exception e)
        {
            EventLog.WriteEntry("CAIT0 Service", e.Message,
                                EventLogEntryType.Warning);
        }
        return myServerName;
    }

    public static String getRefFileName(RegistryKey regkey)
    {
        // Get the name of the instruction file
        String myRefFileName = null;
        try
        {
            myRefFileName = regkey.GetValue("RefFileName").ToString();
        }
        catch (Exception e)
        {
        }
    }
}
EventLog.WriteEntry("CaitO Service", e.Message, EventLogEntryType.Warning);
}

return myRefFileName;
}
/*
* Name: Jason Batchelor
* Program: CAIT
* Date: 10/3/2010
*
* This program is a proof of concept being done for the partial fulfillment
of the MS in Networking and Systems
* Administration from the Rochester Institute of Technology. All rights
reserved.
*
* ThreadAggregation.cs: Handles all accessor methods for information about
each thread.
*/

using System;
using System.Diagnostics;

public class ThreadAggregation
{
    public int threadID;
    public DateTime startTime;
    public TimeSpan cpuTime;
    public int priority;
    public IntPtr startAddress;

    public ThreadAggregation(ProcessThread pt)
    {
        threadID = getThreadID(pt);
        startTime = getStartTime(pt);
        cpuTime = getCpuTime(pt);
        priority = getPriority(pt);
        startAddress = getStartAddress(pt);
    }

    public static int getThreadID(ProcessThread pt)
    {
        try { return pt.Id; }
        catch (Exception) { return 0; }
    }

    public static DateTime getStartTime(ProcessThread pt)
    {
        try { return pt.StartTime; }
        catch (Exception) { return DateTime.Now; }
    }

    public static TimeSpan getCpuTime(ProcessThread pt)
    {
        try { return pt.TotalProcessorTime; }
        catch (Exception) { return TimeSpan.Zero; }
    }

    public static int getPriority(ProcessThread pt)
    {

try { return pt.BasePriority; }
catch (Exception) { return 0; }

public static IntPtr getStartAddress(ProcessThread pt)
{
    try { return pt.StartAddress; }
    catch (Exception) { return IntPtr.Zero; }
}

namespace WindowsService
{
    [RunInstaller(true)]
    public class WindowsServiceInstaller : Installer
    {
        // Constructor for the windows service installer
        public WindowsServiceInstaller()
        {
            var processInstaller = new ServiceProcessInstaller();
            var serviceInstaller = new ServiceInstaller();

            // Install service with the local system account
            processInstaller.Account = ServiceAccount.LocalSystem;

            // Set the service display name of the service
            serviceInstaller.DisplayName = "CaitO";
            serviceInstaller.StartType = ServiceStartMode.Automatic;

            // Set the actual service name and description
            serviceInstaller.ServiceName = "CaitO";
            serviceInstaller.Description = "Aggregation service for host computing node"

            this.Installers.Add(processInstaller);
            this.Installers.Add(serviceInstaller);
        }
    }
}
using System;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Data.SqlClient;
using System.Data.OleDb;

public partial class _Default : System.Web.UI.Page
{
    protected void Button1_Click(object sender, EventArgs e)
    {
        String queryText = TextBox1.Text; // get analyst query text
        String queryTable = DropDownList1.SelectedValue; // get dropdown box query

        String dbServer = "192.168.1.8";
        String dbName = "SIMS";
        String serverUID = "SIMSCli";
        String serverPwd = "vmware";
        // create connection object
        OleDbConnection conn = new OleDbConnection("Provider=sqloledb;server=" + dbServer + ";" +
                                                  "database=" + dbName + ";" +
                                                  "uid=" + serverUID + ";" +
                                                  "pwd=" + serverPwd);

        DataSet ds = new DataSet();
        // craft db query based on analyst input and selections
        // output to web page the query results
        try
{  
    conn.Open();  
    if (queryTable.Equals("Snapshot ID"))  
    {  
        OleDbDataAdapter da = new OleDbDataAdapter("select SnapID, SnapDate, ClientMAC, IPAddress, Hostname from Snapshots where SnapID LIKE "+ queryText + ", conn);  
        da.Fill(ds, "Snapshots");  
        GridView1.DataSource = ds.Tables["Snapshots"].DefaultView;  
        GridView1.DataBind();  
    }  
    if (queryTable.Equals("Directory ID"))  
    {  
        OleDbDataAdapter da = new OleDbDataAdapter("select DirectoryID, SnapID, DirectoryName from Directories where DirectoryID LIKE "+ queryText + ", conn);  
        da.Fill(ds, "Directories");  
        GridView1.DataSource = ds.Tables["Directories"].DefaultView;  
        GridView1.DataBind();  
    }  
    if (queryTable.Equals("FileName"))  
    {  
        OleDbDataAdapter da = new OleDbDataAdapter("select DirectoryID, FileID, FileName, LastWriteTime, LastAccessTime, FileCreationTime, FileSize, FileMD5, FileADS from Files where FileName LIKE "+ queryText + ", conn);  
        da.Fill(ds, "Files");  
        GridView1.DataSource = ds.Tables["Files"].DefaultView;  
        GridView1.DataBind();  
    }  
    if (queryTable.Equals("FileMD5"))  
    {  
        OleDbDataAdapter da = new OleDbDataAdapter("select DirectoryID, FileID, FileName, LastWriteTime, LastAccessTime, FileCreationTime, FileSize, FileMD5, FileADS from Files where FileMD5 LIKE "+ queryText + ", conn);  
        da.Fill(ds, "Files");  
        GridView1.DataSource = ds.Tables["Files"].DefaultView;  
        GridView1.DataBind();  
    }  
    if (queryTable.Equals("IP Address"))  
    {  
        OleDbDataAdapter da = new OleDbDataAdapter("select SnapID, LocalTCPConnection, RemoteTCPConnection, TcpConnectionState from ActiveTcpConnections where LocalTCPConnection Like "+ queryText + ", conn);  
        da.Fill(ds, "ActiveTcpConnections");  
        GridView1.DataSource = ds.Tables["ActiveTcpConnections"].DefaultView;  
        GridView1.DataBind();  
    }  
    if (queryTable.Equals("AdsData"))  
    {  
        OleDbDataAdapter da = new OleDbDataAdapter("select FileID, ADSPath, ADSSize from AdsData where ADSPath Like "+ queryText + ", conn);  
        da.Fill(ds, "AdsData");  
    }  
}
GridView1.DataSource = ds.Tables["AdsData"].DefaultView;
GridView1.DataBind();

if (queryTable.Equals("FileSize"))
{
    OleDbDataAdapter da = new OleDbDataAdapter("select DirectoryID, FileID, FileName, LastWriteTime, LastAccessTime,
    FileCreationTime, FileSize, FileMD5, FileADS from Files where FileSize LIKE " + queryText + "'", conn);
    da.Fill(ds, "Files");
    GridView1.DataSource = ds.Tables["Files"].DefaultView;
    GridView1.DataBind();
}

if (queryTable.Equals("ProcessName"))
{
    OleDbDataAdapter da = new OleDbDataAdapter("select SnapID,
    ProcID, PID, ProcessName, ProcessOwner, StartTime, CPUTime, PhysicalMemSize,
    Threads from ProcessList where ProcessName LIKE " + queryText + "'", conn);
    da.Fill(ds, "ProcessList");
    GridView1.DataSource = ds.Tables["ProcessList"].DefaultView;
    GridView1.DataBind();
}

if (queryTable.Equals("Threads"))
{
    OleDbDataAdapter da = new OleDbDataAdapter("select StartTime,
    CPUTime, Priority, StartAddress from ThreadList where ProcID LIKE " + queryText + "'", conn);
    da.Fill(ds, "ThreadList");
    GridView1.DataSource = ds.Tables["ThreadList"].DefaultView;
    GridView1.DataBind();
}

// close the connection
finally
{
    if (conn != null)
    {
        conn.Close();
    }
}
}
This web template is a proof of concept being done for the partial fulfillment of the MS in Networking and Systems Administration from the Rochester Institute of Technology. All rights reserved.

default.aspx: Web page with some embedded server side calls for database queries.

```csharp
%@ Page Language="C#" AutoEventWireup="true" CodeFile="default.aspx.cs" Inherits="_Default" %>

@ Register assembly="System.Web.DataVisualization, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"
namespace="System.Web.UI.DataVisualization.Charting" tagprefix="asp" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<link type="text/css" rel="Stylesheet" href="Style.css" />
<title>Client Aggregation Command Center</title>
<style type="text/css">
.style1
{
    width: 787px;
}
.style4
{
    width: 449px;
    text-align: center;
    height: 59px;
    font-family: Arial, Helvetica, sans-serif;
    font-size: small;
}
.style5
{
    text-align: center;
}
.style7
{
    font-family: Arial, Helvetica, sans-serif;
}
.style10
{
    font-family: Arial, Helvetica, sans-serif;
    font-size: small;
    font-weight: bold;
}
```
Client Aggregation Command Center

Summary:

<table>
<thead>
<tr>
<th>Field</th>
<th>DetailsView ID</th>
<th>AutoGenerateRows</th>
<th>DataSourceID</th>
<th>Height</th>
<th>Width</th>
<th>BorderColor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reporting Clients:</td>
<td>DetailsView1</td>
<td>False</td>
<td>Num_Clients</td>
<td>50px</td>
<td>170px</td>
<td>White</td>
</tr>
<tr>
<td></td>
<td>SqlDataSource</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SqlDataSource</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
<RowStyle BorderColor="Black" />
<Fields>
  <asp:BoundField DataField="Column1" HeaderText="ADS Count:" ReadOnly="True" SortExpression="Column1" />
</Fields>
</asp:DetailsView>
<asp:SqlDataSource ID="ADS_Num" runat="server"
ConnectionString="Data Source=192.168.1.8;InitialCatalog=SIMS;User ID=SIMSCli;Password=vmware"
SelectCommand="SELECT COUNT(*) FROM AdsData"></asp:SqlDataSource>
</td>
</tr>
<tr class="style10">
  <asp:DetailsView ID="DetailsView2" runat="server"
AutoGenerateRows="False"
DataSourceID="Snapshot_Number" Height="50px"
Width="170px"
BorderColor="White">
  <Fields>
    <asp:BoundField DataField="Column1" HeaderText="Snapshot Count:" ReadOnly="True"
SortExpression="Column1" />
  </Fields>
</asp:DetailsView>
<asp:SqlDataSource ID="Snapshot_Number" runat="server"
ConnectionString="Data Source=192.168.1.8;InitialCatalog=SIMS;User ID=SIMSCli;Password=vmware"
SelectCommand="SELECT COUNT(*) FROM Snapshots"></asp:SqlDataSource>
</td>
</tr>
<tr class="style10">
  <asp:DetailsView ID="DetailsView4" runat="server"
AutoGenerateRows="False"
DataSourceID="Last_Report" Height="50px" Width="300px"
BorderColor="White">
  <RowStyle BorderColor="White" />
  <Fields>
    <asp:BoundField DataField="SnapDate" HeaderText="Last Check-In:" SortExpression="SnapDate" HeaderText="Last Check-In:" />
  </Fields>
</asp:DetailsView>
<asp:SqlDataSource ID="Last_Report" runat="server"
ConnectionString="Data Source=192.168.1.8;InitialCatalog=SIMS;User ID=SIMSCli;Password=vmware"
SelectCommand="SELECT TOP 1 SnapDate FROM Snapshots ORDER BY SnapID DESC"></asp:SqlDataSource>
</td>
</tr>
<tr class="style10">
  <asp:Chart ID="Chart1" runat="server"
DataSourceID="Popular_Connections">
<Titles>
  <asp:Title Font="Microsoft Sans Serif, 9.75pt, style=Bold"
    Name="Popular Connections" Alignment="TopCenter"
    DockedToChartArea="Popular Connections"
    IsDockedInsideChartArea="False"
    Text="Popular Connections">
    <asp:Title Alignment="BottomCenter"
      DockedToChartArea="Popular Connections"
      Docking="Bottom" IsDockedInsideChartArea="False"
      Name="Connections"
      Text="Connections" TextOrientation="Horizontal">
      </asp:Title>
  </Titles>
</Titles>

<series>
  <asp:Series Name="Series1" ChartArea="Popular Connections"
    ChartType="Bar" Legend="Legend1" XValueMember="IPAddress"
    YValueMembers="Occurrences">
    </asp:Series>
  </series>
</charts>

<asp:SqlDataSource ID="Popular_Connections" runat="server"
  ConnectionString="Data Source=192.168.1.8;Initial Catalog=SIMS;User ID=SIMSCli;Password=vmware"
  SelectCommand="SELECT TOP 5 SUBSTRING(RemoteTCPConnection,0,CHARINDEX(':',RemoteTCPConnection)) AS IPAddress,
    COUNT(SUBSTRING(RemoteTCPConnection,0,CHARINDEX(':',RemoteTCPConnection))) AS Occurrences FROM ActiveTcpConnections GROUP BY SUBSTRING(RemoteTCPConnection,0,CHARINDEX(':',RemoteTCPConnection)) HAVING SUBSTRING(RemoteTCPConnection,0,CHARINDEX(':',RemoteTCPConnection)) NOT LIKE '127.0.0.1' ORDER BY Occurrences DESC">
  </asp:SqlDataSource>
</td>
</td class="style4">
<asp:Chart ID="Chart2" runat="server"
  DataSourceID="Clients_Last12Hrs">
  <Titles>
    <asp:Title Alignment="TopCenter"
      DockedToChartArea="Reports In Last 12hrs" Font="Microsoft Sans Serif, 9.75pt, style=Bold"
      IsDockedInsideChartArea="False"
      Name="Reports In Last 12hrs" Text="Reports In Last 12hrs">
      </asp:Title>
  </Titles>
  <asp:Title Alignment="BottomCenter"
    DockedToChartArea="Reports In Last 12hrs"
Docking="Bottom" IsDockedInsideChartArea="False"
Name="Hours" Text="Hour"
  TextOrientation="Horizontal">
  </asp:Title>
  <asp:Title DockedToChartArea="Reports In Last 12hrs"
Docking="Left"
Text="Reports"
  TextOrientation="Rotated270">
  </asp:Title>
</Titles>
<Series>
  <asp:Series Name="Series1" XValueMember="Hour"
YValueMembers="Total"
  ChartArea="Reports In Last 12hrs">
  </asp:Series>
</Series>
</Charts>
<asp:ChartArea Name="Reports In Last 12hrs">
</asp:ChartArea>
<asp:SqlDataSource ID="Clients_Last12Hrs" runat="server"
ConnectionString="Data Source=192.168.1.8;InitialCatalog=SIMS;User ID=SIMSCli;Password=vmware"
SelectCommand="SELECT DATEPART(hh,SnapDate) AS Hour,
COUNT(DATEPART(hh,SnapDate)) AS Total FROM (SELECT SnapDate FROM SnapShots
WHERE SnapDate &gt;= DATEADD(hour, -12, GETDATE()) AND SnapDate &lt;=
GETDATE()) AS Dates GROUP BY DATEPART(hh,SnapDate)">
</asp:SqlDataSource>
  </tr>
</table>
</td>
</tr>
<tr>
  <td class="style5" colspan="2">
    &nbsp;</td>
</tr>
<tr>
  <td class="style10" colspan="2">
    <asp:Button ID="Search" runat="server" Text="Search"
onclick="Button1_Click" />
    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n...
</td>
</tr>
<tr>
<td class="style10" colspan="2">
<asp:GridView ID="GridView1" runat="server">
</asp:GridView>
</td>
</tr>
</table>
</form>
</body>
</html>
Bibliography

   <http://technology.timesonline.co.uk/tol/news/tech_and_web/article7035997.ece>

   <http://www.post-gazette.com/pg/09364/1024438-298.stm>


   <http://searchsecurity.techtarget.com/magazineFeature/0,296894,sid14_gci1313260,00.html>


   <http://doi.acm.org/10.1145/1113034.1113068>


